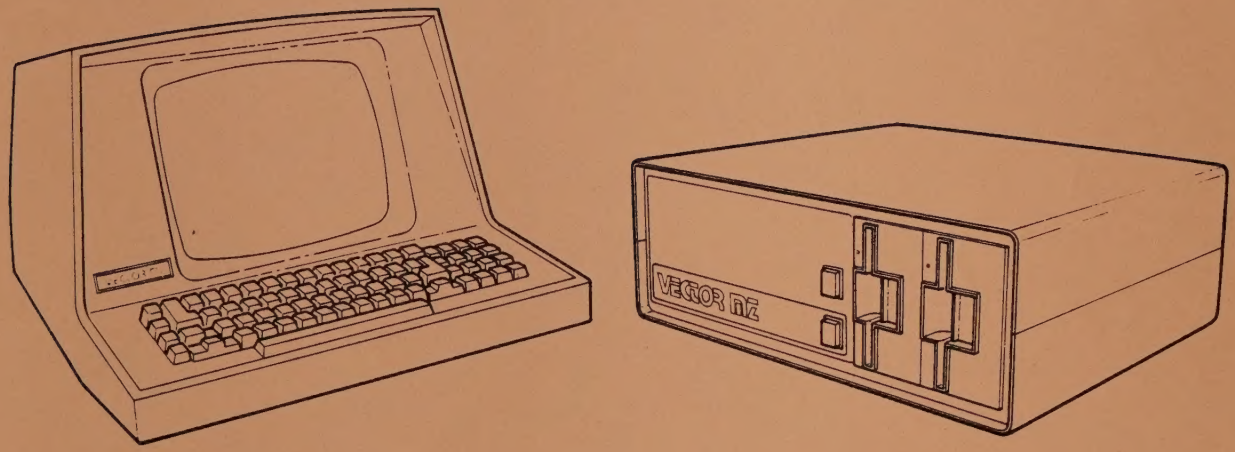
BITSTREAMER I

USERS MANUAL



VECTOR GRAPHIC INC.

BIT STREAMER I/O BOARD

TABLE OF CONTENTS

SUBJECT	PAGE
INTRODUCTION	1
PARTS LISTS.	2
SCHEMATIC	3A
COMPONENT DIAGRAM	3B
THEORY OF OPERATION	4
USERS GUIDE	6
APPENDIX - CUSTOM APPLICATIONS	14
GENERAL TROUBLE SHOOTING GUIDE	15

INTRODUCTION

VECTOR GRAPHIC INC.'S "BIT STREAMER" I/O BOARD IS DESIGNED FOR EASE OF OPERATION AND MAXIMUM FLEXIBILITY. AN 8251 USART IS EMPLOYED, WHICH UNDER SOFTWARE CONTROL IS CAPABLE OF A WIDE VARIETY OF DATA CONFIGURATIONS. THE SERIAL PORT INCLUDES DRIVERS AND RECEIVERS FOR BOTH RS232C OR 20MA CURRENT LOOP OPERATION. INCLUDED IN THIS MANUAL ARE DIRECTIONS FOR A SIMPLE MODIFICATION TO PERMIT TWO SERIAL PORTS.

THE TWO PARALLEL I/O PORTS FEATURE LATCHED OUTPUT DATA LINES AND LATCHES FOR THE INPUT STROBES.

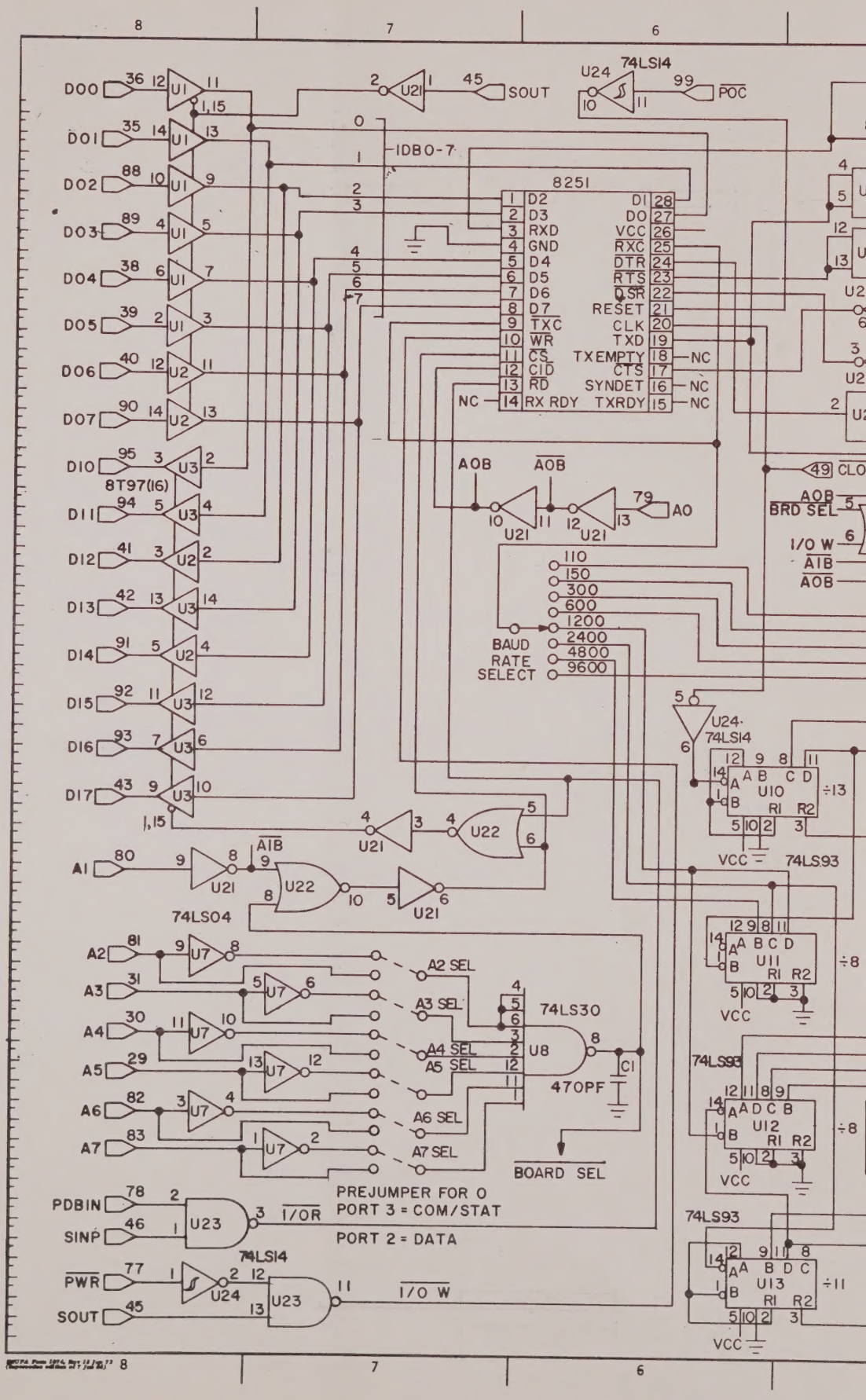
CAREFUL ATTENTION TO GOOD DESIGN PRACTICES PROVIDES THE USER WITH AN I/O BOARD THAT OPERATES RELIABLY WITHOUT CONSTANT ADJUSTMENT OR MAINTENANCE.

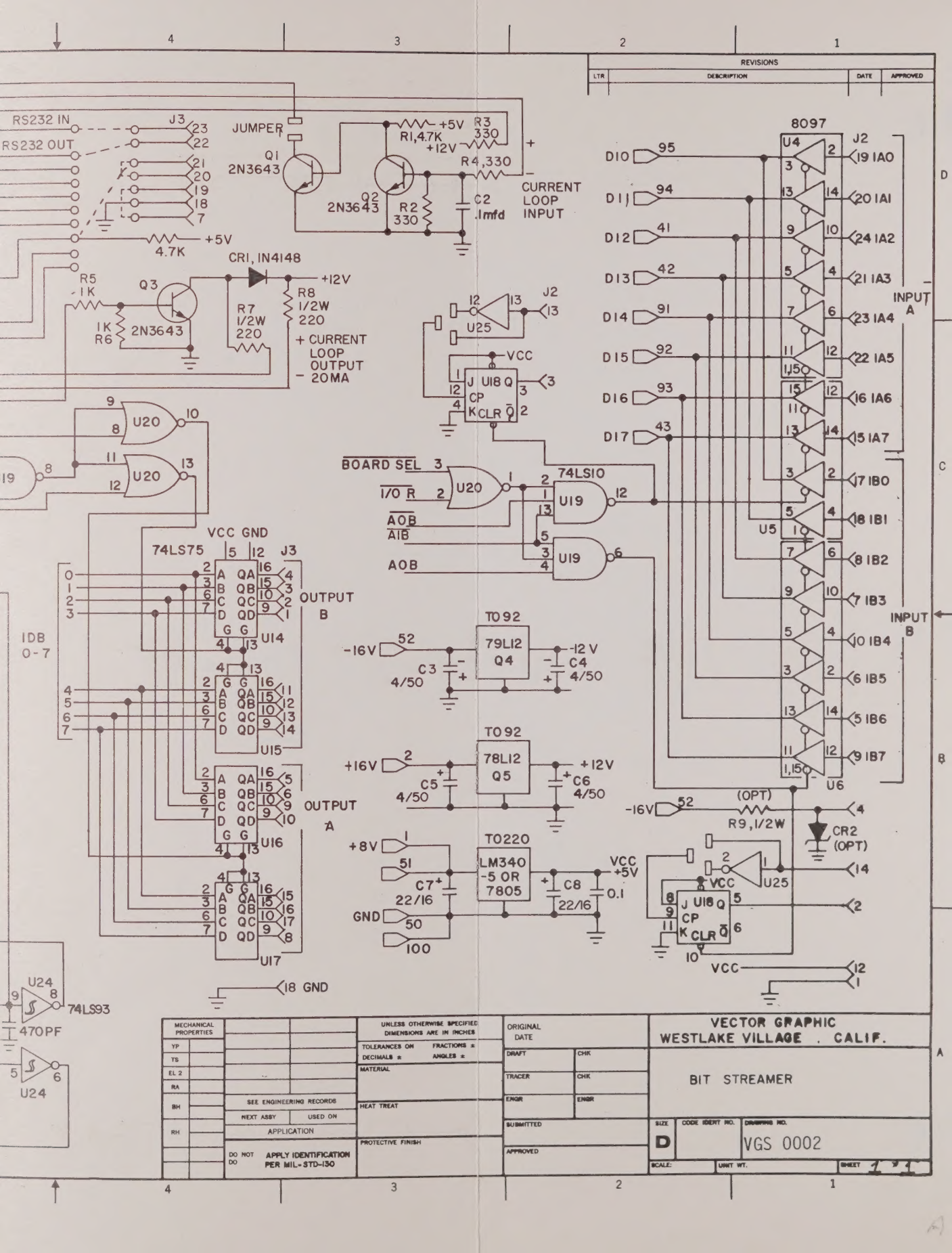
IF THERE IS ANYTHING YOU DO NOT UNDERSTAND, PLEASE DO NOT HESITATE TO CALL OR WRITE US.

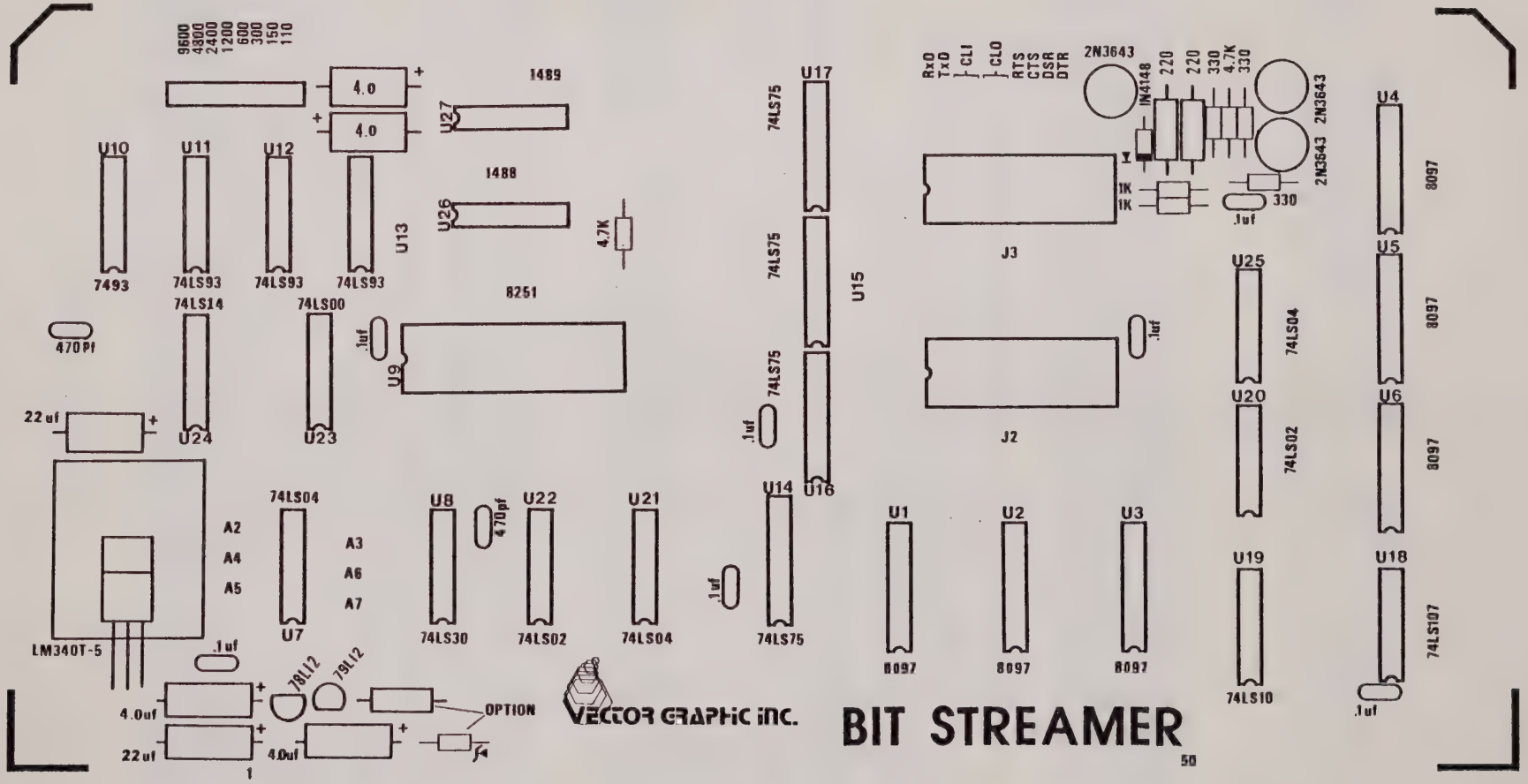
PARTS LISTS

QTY.	DESCRIPTION
1	PRINTED CIRCUIT BOARD
1	6106B HEAT SINK
1	7805/340T-5 REGULATOR
1	78L12
1	79L12
1	8251 USART (U9)
1	7493 (U10)
3	74LS93 (U11, U12, U13)
6	74367/8097 (U1, U2, U3, U4, U5, U6)
1	74LS30 (U8)
2	74LS02 (U20, U22)
1	74LS00 (U23)
4	74LS75 (U14, U15, U16, U17)
1	74LS14 (U24)
3	74LS04 (U7, U21, U25)
1	74LS107 (U18)
1	74LS10 (U19)
1	1488 LINE DRIVER (U26)
1	1489 LINE RECEIVER (U27)
2	470 PF 50 VOLT AXIAL GLASS CAPACITORS
7	0.1 MFD 50V MONOLITHIC RADIAL CAPACITORS
4	4.0 MFD 50V AXIAL ELECTROLYTIC CAPACITORS
2	22 MFD 16V AXIAL ELECTROLYTIC CAPACITORS

QTY.	DESCRIPTION
3	330 OHM 1/4 WATT CARBON RESISTORS (STRIPES OF ORANGE, ORANGE, BROWN)
2	220 OHM 1/2 WATT CARBON RESISTORS (STRIPES OF RED, RED, BROWN)
2	4.7K 1/4 WATT RESISTOR (STRIPES OF YELLOW, VIOLET, RED)
2	1K 1/4 WATT RESISTORS (STRIPES OF BROWN, BLACK, RED)
3	2N3643 TRANSISTORS
1	1N4148 DIODE
1	MOLEX PLUG
2	24 PIN DIP PLUGS
1	8 POSITION DIP SWITCH
2	24 PIN SOCKETS
10	16 PIN SOCKETS
16	14 PIN SOCKETS
1	28 PIN SOCKET
2	EJECTORS
OPTIONAL	
1	CABLE #ASI-1327


8
7
6
D00
D01
D02
D03
D04
D05
D06
D07
D10
D11
D12
D13
D14
D15
D16
D17
8T97(16)
U1
U2
U3
1,15
U21
SOUT
U24
74LS14
POC
IDB0-7
8251
D2
D3
RXD
GND
D4
D5
D6
D7
TXC
WR
CS
C/D
RD
RX RDY
D1
DO
VCC
RXC
DTR
RTS
DSR
RESET
CLK
TXD
TXEMPTY
CTS
SYNDET
TXRDY
NC
A0B
A0
BAUD
RATE
SELECT
110
150
300
600
1200
2400
4800
9600
BRD SEL
I/O W
A1B
U22
A1
74LS04
A2
A3
A4
A5
A6
A7
U7
A2 SEL
A3 SEL
A4 SEL
A5 SEL
A6 SEL
A7 SEL
74LS30
U8
470PF
C1
BOARD SEL
PREJUMPER FOR 0
PORT 3 = COM/STAT
PORT 2 = DATA
PDBIN
SINP
PWR
SOUT
U23
I/OR
I/O W
U10
U11
U12
U13
÷13
÷8
÷11
74LS93
VCC


REVISIONS
LTR
DESCRIPTION
DATE
APPROVED
RS232 IN
RS232 OUT
J3
JUMPER
Q1
2N3643
Q2
2N3643
R1,4.7K
+5V
R3
330
+12V
R4,330
R2
330
C2
.1mfd
CURRENT
LOOP
INPUT
4.7K
+5V
CR1, IN4148
R5
1K
Q3
1K
R6
2N3643
R7
1/2W
220
R8
1/2W
220
+12V
+ CURRENT
LOOP
OUTPUT
- 20MA
8097
U4
J2
D10
D11
D12
D13
D14
D15
D16
D17
19 IA0
20 IA1
24 IA2
21 IA3
23 IA4
22 IA5
16 IA6
15 IA7
INPUT
A
U25
VCC
U18
J
K
CP
CLR
Q
U20
BOARD SEL
I/O R
AOB
AIB
74LS10
U19
7 IB0
8 IB1
8 IB2
7 IB3
10 IB4
6 IB5
5 IB6
9 IB7
INPUT
B
U5
U6
VCC GND
74LS75
J3
OUTPUT
B
IDB
0-7
U14
U15
OUTPUT
A
U16
U17
18 GND
TO92
79L12
Q4
-16V
-12 V
C3
4/50
C4
4/50
TO92
78L12
Q5
+16V
+12V
C5
4/50
C6
4/50
TO220
LM340
-5 OR
7805
+8V
VCC
+5V
C7
22/16
C8
22/16
0.1
GND
50
100
(OPT)
R9,1/2W
CR2
(OPT)
U24
74LS93
470PF
MECHANICAL
PROPERTIES
YP
TS
EL 2
RA
BH
RH
SEE ENGINEERING RECORDS
NEXT ASSY
USED ON
APPLICATION
DO NOT
DO
APPLY IDENTIFICATION
PER MIL-STD-130
UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES ON
FRACTIONS ±
DECIMALS ±
ANGLES ±
MATERIAL
HEAT TREAT
PROTECTIVE FINISH
ORIGINAL
DATE
DRAFT
CHK
TRACER
CHK
ENGR
ENGR
SUBMITTED
APPROVED
VECTOR GRAPHIC
WESTLAKE VILLAGE . CALIF.
BIT STREAMER
SIZE
D
CODE IDENT NO.
DRAWING NO.
VGS 0002
SCALE:
UNIT WT.
SHEET


9600
4800
2400
1200
600
300
150
110
4.0
4.0
1489
1488
U27
U26
U13
4.7K
U10
U11
U12
7493
74LS93
74LS93
74LS93
74LS14
74LS00
470Pf
.1uf
U9
8251
22 uf
U24
U23
A2
A4
A5
A3
A6
A7
74LS04
U7
U8
470pf
U22
U21
74LS30
74LS02
74LS04
LM340T-5
.1uf
78L12
79L12
OPTION
4.0uf
22 uf
4.0uf
1
U17
74LS75
74LS75
74LS75
U15
.1uf
U14
U16
74LS75
.1uf
RxD
TxD
CLI
CLO
RTS
CTS
DSR
DTR
2N3643
IN4148
220
220
330
4.7K
330
2N3643
2N3643
1K
1K
330
.1uf
J3
J2
.1uf
U25
74LS04
U20
74LS02
U19
74LS10
U1
U2
U3
8097
8097
8097
U4
U5
U6
U18
8097
8097
8097
74LS107
.1uf
VECTOR GRAPHIC INC.
BIT STREAMER
50

THEORY OF OPERATION

THE BIT STREAMER COMBINES TWO PARALLEL INPUT PORTS, TWO PARALLEL OUTPUT PORTS AND A SERIAL I/O PORT USING AN 8251 PROGRAMMABLE UNIVERSAL SYNCHRONOUS / ASYNCHRONOUS RECEIVER - TRANSMITTER. COMMUNICATION WITH THE CIRCUITRY ON THE BOARD IS BY MEANS OF INPUT AND OUTPUT INSTRUCTION CYCLES EXECUTED BY THE CPU.

INPUT PORTS

TO INPUT DATA, THE FOLLOWING SIGNAL CONDITIONS MUST BE MET:

1. THE LOWER EIGHT ADDRESS LINES MUST CONTAIN THE PORT ADDRESS.

2. SINP MUST BE HIGH, INDICATING AN INPUT MACHINE CYCLE.

3. PDBIN MUST BE HIGH, INDICATING THAT THE DATA BUS MAY BE ACCESSED BY THE BOARD.

THE REQUIRED SIGNALS ARE GATED TOGETHER THROUGH U8, U23, U20 AND U19 TO PRODUCE TWO ENABLE SIGNALS -- ONE FOR PORT A AND ONE FOR PORT B -- WHICH STROBE THE INPUT DATA ONTO THE DATA BUS AT THE APPROPRIATE TIME USING TRI-STATE BUS DRIVERS U4, U5 AND U6.

OUTPUT PORTS

VALID OUTPUT DATA IS PRESENT ON THE DATA BUS DURING AN OUTPUT MACHINE CYCLE WHEN THE FOLLOWING CONDITIONS ARE MET:

1. THE LOWER EIGHT ADDRESS LINES CONTAIN THE PORT ADDRESS.

2. SOUT MUST BE HIGH, INDICATING AN OUTPUT MACHINE CYCLE.

3. PWR MUST BE LOW, INDICATING THE DATA BUS CONTAINS VALID DATA.

THE APPROPRIATE SIGNALS ARE GATED TOGETHER THROUGH U8, U23, U19 AND U20 TO PRODUCE STROBES TO CAUSE THE DATA BUS SIGNALS TO BE LATCHED BY 74LS75 QUAD LATCHES U14, U15, U16 AND U17. THESE OUTPUTS REMAIN CONSTANT UNTIL THE PARTICULAR OUTPUT PORT IS AGAIN ACCESSED BY THE CPU. AN INTERNAL BI-DIRECTIONAL DATA BUS IS USED TO MINIMIZE THE LOADING ON THE S-100 BUS.

SERIAL PORT

THE HEART OF THE SERIAL I/O PORT IS THE 8251 USART CONSISTING OF INDEPENDENT RECEIVER AND TRANSMITTER. THE FUNCTION OF THE TRANSMITTER IS TO ACCEPT EIGHT BITS OF PARALLEL DATA FROM THE DATA BUS, AND CONVERT THIS TO SERIAL DATA WITH A WIDE RANGE OF ASYNCHRONOUS FORMATS. THE SPEED AT WHICH THE DATA AND SYNC BITS

ARE OUTPUT IS CALLED THE BAUD RATE (AFTER BAUDOT) AND THIS IS CONTROLLED BY A SELECTABLE CLOCK RATE DERIVED FROM THE 2 MHZ SYSTEM CLOCK BY COUNTERS U10, U11, U12 AND U13. THE CLOCK RATE IS NORMALLY SIXTEEN TIMES THE REQUIRED BAUD RATE (THIS CAN BE CHANGED WHEN THE USART IS INITIALIZED), SO A FREQUENCY OF 153.6 KHZ IS REQUIRED FOR 9600 BAUD. THIS GIVES A FREQUENCY DIVISION RATIO OF 13.02 (WHICH IS ROUNDED TO 13) RELATIVE TO THE 2 MHZ CLOCK. U10 THUS FORMS A DIVIDE BY 13 STAGE, AND THE GATED TERMINAL COUNT AT U23 PIN 8 CAN BE SELECTED BY A DIP SWITCH FOR 9600 BAUD. THE COMMON BAUD RATES, EXCEPT FOR 110 BAUD, ARE OBTAINED BY SUCCESSIVELY DIVIDING THIS FREQUENCY IN HALF USING U11 AND U12 (74LS93 BINARY COUNTERS). THE CLOCK FREQUENCY FOR 110 BAUD, THE STANDARD FREQUENCY FOR TELETYPES, IS GENERATED BY DIVIDING THE FREQUENCY FOR 2400 BAUD BY 22. THESE FREQUENCIES ARE SELECTED BY THE DIP SWITCH AND APPLIED TO TXC AND RXC OF THE USART.

THE USART IS DESIGNED TO INTERFACE EASILY TO AN 8080 BUS STRUCTURE, AND THE CONTROL SIGNALS RD, C/D, CS, WR ARE DERIVED EASILY FROM THE S-100 BUS SIGNALS. SINCE THE DEVICE WAS INTENDED TO BE USED WITH A BI-DIRECTIONAL DATA BUS, THE S-100 SPLIT DATA BUS IS RECOMBINED AS AN INTERNAL DATA BUS USING THE TRI-STATE BUS DRIVERS U1, U2 AND U3. EIA RS232C LEVEL LINE DRIVERS AND RECEIVERS U27 AND U26 INTERFACE THE RXD INPUT AND TXD OUTPUT OF THE USART TO THE OUTSIDE WORLD. THESE COMPONENTS ARE CAPABLE OF TOLERATING MORE ABUSE IN THEIR CIRCUITS AND STATIC TRANSIENTS THAN ARE TTL OR MOS DEVICES, AND PRODUCE PLUS OR MINUS 10 VOLT SIGNALS WITH HIGH NOISE IMMUNITY. IN ADDITION, DISCRETE COMPONENTS ARE USED TO INTERFACE WITH 20MA TELETYPE SIGNALS.

THE USART IS CAPABLE OF BEING CONFIGURED FOR A WIDE VARIETY OF COMMUNICATION FORMATS. THE EXACT SOFTWARE REQUIRED TO COVER PERHAPS 90% OF THE APPLICATIONS FOR PERSONAL OR BUSINESS COMPUTERS IS DESCRIBED IN THE USERS GUIDE. FOR MORE SOPHISTICATED APPLICATIONS, THE USER IS REFERRED TO THE INTEL PUBLICATIONS AP-16 AND 1977 DATA CATALOG OR ADAM OSBORNE'S "AN INTRODUCTION TO MICROCOMPUTERS" VOLUME II.

USERS GUIDE

THE BIT STREAMER I/O BOARD HAS BEEN DESIGNED TO MINIMIZE THE NEED FOR USER INSTALLED JUMPERS. WITHOUT MAKING ANY CHANGES TO THE PRE-JUMPERED OPTIONS, THE BOARD CAN BE INSTALLED IN A COMPUTER AND WILL OPERATE AS AN RS232 SERIAL PORT USING THE INITIALIZATION AND I/O SOFTWARE ON THE VECTOR GRAPHIC OPTION C MONITOR PROM. CONNECTION BETWEEN THE BOARD SOCKET J3 AND DB25 TERMINAL CONNECTOR CAN BE MADE BY MEANS OF A 24 WIRE FLAT RIBBON CABLE WITH A 24 PIN DIP PLUG ON ONE END AND A DB25 FEMALE CONNECTOR ON THE OTHER. THIS CABLE IS AVAILABLE FROM VECTOR GRAPHIC AS AN OPTION. THE PINOUT HAS BEEN CONFIGURED SO THAT THE RS232 DATA AND CONTROL LINES WILL MATCH; PIN 24 ON THE DIP PLUG CONNECTS TO PIN 1 ON THE DB25. THE PARALLEL OUTPUT PORTS ARE AVAILABLE ON THE SAME CONNECTOR, WHILE THE PARALLEL INPUT PORTS ARE AVAILABLE AT J2. FIGURE 1 GIVES THE PINOUTS FOR THE VARIOUS LINES.

IN THE STANDARD CONFIGURATION, THE BOARD OCCUPIES FOUR I/O PORT ADDRESSES AS SHOWN BELOW:

PORT	INPUT	OUTPUT
03	USART STATUS	USART COMMAND
02	USART DATA	USART DATA
01	PARALLEL PORT B	PARALLEL PORT B
00	PARALLEL PORT A	PARALLEL PORT A

THE PARALLEL PORT OPERATION IS QUITE STRAIGHTFORWARD: AN "OUT 0" INSTRUCTION EXECUTED BY THE CPU WILL CAUSE THE 8 BIT CONTENTS OF THE ACCUMULATOR TO BE LATCHED ON THE BOARD AND THE TTL LEVEL SIGNALS CORRESPONDING TO THE VARIOUS BITS WILL BE AVAILABLE AT PORT A AS SHOWN IN FIGURE 1. SIMILARLY, AN "IN 1" INSTRUCTION WILL CAUSE THE SIGNALS APPLIED TO PORT B ON J3 TO BE SAMPLED AND TRANSFERRED TO THE ACCUMULATOR. INPUT VOLTAGES SHOULD NOT EXCEED THE RANGE OF 0-5V, AND THE OUTPUTS ARE ACTIVE PULL-UP LOW POWER SCHOTTKY GATES WITH A FAN OUT OF FIVE STANDARD TTL GATES OR 20 LOW POWER SCHOTTKY GATES.

OPERATION OF THE SERIAL PORT IS CONSIDERABLY MORE COMPLEX AND WILL BE DESCRIBED IN MORE DETAIL ELSEWHERE. BRIEFLY, THE 8251 USART MUST BE CONFIGURED BY OUTPUTTING TWO COMMANDS TO PORT 3 ONCE, AND ONLY ONCE, EACH TIME THE COMPUTER IS TURNED ON OR RESET. THE VECTOR GRAPHIC CPU BOARD GENERATES A POC SIGNAL IN EITHER CASE. THE INITIALIZATION ROUTINE FOR MOST APPLICATIONS IS AS FOLLOWS:

```
INIT    MVI A, 0CEH
        OUT 3
        MVI A, 27H
        OUT 3
```

THE FIRST COMMAND CONFIGURES THE USART FOR ASYNCHRONOUS OUTPUT WITH A BAUD RATE CLOCK DIVIDE FACTOR OF 16, AN 8 BIT CHARACTER LENGTH, NO PARITY BIT AND TWO STOP BITS. THIS IS THE STANDARD FORMAT FOR 110 BAUD COMMUNICATION, BUT IT WILL WORK

SATISFACTORILY AT HIGHER BAUD RATES (I.E. WITH AN ADM-3 TERMINAL), ALTHOUGH ONLY ONE STOP BIT IS NORMALLY USED. THE EXTRA STOP BIT WILL RESULT IN A 10% TRANSMISSION SPEED REDUCTION, AND APPEARS TO THE TERMINAL AS A BRIEF MARK CONDITION BETWEEN CHARACTERS. THE SECOND COMMAND ENABLES THE TRANSMITTER AND RECEIVER TO BEGIN DATA TRANSMISSION.

RECEIPT OF A CHARACTER IS INDICATED BY TESTING THE STATUS PORT. A TYPICAL INPUT ROUTINE IS AS FOLLOWS:

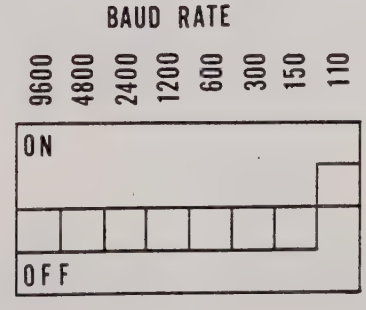
```
INPUT   IN      3       USART STATUS
        XRI     00
        ANI     2       LOOK AT D1 - RXRDY
        JZ      INPUT   LOOP UNTIL TRUE
        IN      2       READ DATA
        ANI     7FH     MASK EIGHTH BIT
        RET
```

TRANSMISSION OF A CHARACTER STARTS BY TESTING THE STATUS PORT TO SEE IF THE USART IS READY TO RECEIVE A CHARACTER.

```
OUTPUT  IN      3       USART STATUS
        XRI     00
        ANI     1       LOOK AT D0 - TXRDY
        JZ      OUTPT   LOOP UNTIL TRUE
        POP     PSW     (OR MOV A,B)
        OUT     2       OUTPUT DATA
        RET
```

OF COURSE, PRIOR TO CALLING THE OUTPUT ROUTINE, THE CHARACTER CODE TO BE OUTPUT WAS SAVED ON THE STACK OR IN ONE OF THE CPU REGISTERS.

THE BAUD RATE IS CONTROLLED BY BOTH THE BAUD RATE FACTOR SELECTED DURING INITIALIZATION, WHICH FIXES THE INTEGER BY WHICH THE USART DIVIDES THE EXTERNALLY SUPPLIED CLOCK, AND THE BAUD RATE SWITCH SETTING ON THE UPPER LEFT CORNER OF THE BOARD. THE LEFTMOST SWITCH WILL SELECT 9600 BAUD, WHILE THE RIGHTMOST SWITCH SELECTS 110 BAUD. THE X16 BAUD RATE FACTOR IS TAKEN INTO ACCOUNT. ONLY ONE SWITCH SHOULD BE ON AT ANY ONE TIME, AND THE BAUD RATE OF THE TERMINAL MUST AGREE WITH THE SWITCH SETTING.



CONNECTION TO THE TERMINAL

EITHER RS232 OR 20MA CURRENT LOOP SIGNALS CAN BE INTERFACED TO THE BIT STREAMER. THE SIMPLEST METHOD IS TO USE A FLAT RIBBON CABLE AS MENTIONED PREVIOUSLY. CABLES ARE AVAILABLE AT YOUR LOCAL COMPUTER STORE, OR YOU MAY ORDER FROM VECTOR GRAPHIC.

THE INTERCONNECTIONS BETWEEN THE PINS OF J3 ARE SHOWN IN FIGURE 2.

THE CORRESPONDENCE BETWEEN J3 AND THE DB25 CONNECTOR PINS IS AS FOLLOWS:

J3	DB25	RS232 DEFINITION RELATIVE TO TERMINAL
24	1	PROTECTIVE GROUND
23	2	TRANSMITTED DATA (TXD)
22	3	RECEIVED DATA (RXD)
21	4	REQUEST TO SEND (RTS)
20	5	CLEAR TO SEND (CTS)
19	6	DATA SET READY (DSR)
18	7	SIGNAL GROUND (GND)
7	20	DATA TERMINAL READY (DTR)

AS FAR AS THE BIT STREAMER IS CONCERNED, THE ONLY SIGNAL NEEDED FOR PROPER OPERATION IS CLEAR TO SEND (CTS), WHICH MUST BE HIGH TO ENABLE THE USART. IF NO CONNECTION IS MADE, A PULL-UP RESISTOR HOLDS THIS LINE HIGH.

FOR PROPER OPERATION OF THE TERMINAL (IN ADDITION TO SIGNAL GROUND, TRANSMITTED DATA AND RECEIVED DATA) CTS AND DSR MUST BE HELD HIGH. THIS IS DONE BY JUMPERING THEM TO RTS AND DTR RESPECTIVELY, WHICH GO HIGH WHEN THE TERMINAL IS ON. TRACES ON THE BOARD INTERCONNECT THE APPROPRIATE PINS. IF EITHER OF THESE SIGNALS IS NOT AVAILABLE, CTS AND DSR MAY BE CONNECTED TO +5V THROUGH A 4.7K PULLUP RESISTOR.

AN ALTERNATE METHOD OF CONNECTING THE TERMINAL IS TO USE A THREE WIRE CONNECTION TO THE BOARD (RXD, TXD, SIGNAL GROUND) AND CONNECT PIN 4 TO PIN 5 AND PIN 20 TO PIN 6 AT THE TERMINAL REAR PANEL.

IF FULL RS232 PROTOCOL WILL BE USED, CUT THE TRACES AS SHOWN IN FIGURE 3-1 AND ADD THE JUMPERS TO CONNECT THE CONTROL SIGNALS TO THE EIA LINE DRIVERS AND RECEIVERS. THE DSR CONTROL INPUT SIGNAL CAN BE TESTED BY LOOKING AT BIT 7 ON THE STATUS PORT, WHILE THE DTR AND RTS OUTPUTS CAN BE CONTROLLED BY OUTPUTTING THE APPROPRIATE COMMAND TO PORT 3. THESE SIGNALS ARE NOT SUPPORTED BY MOST OF THE SOFTWARE AVAILABLE TO THE HOBBYIST, AND SHOULD BE CONNECTED ONLY IN EXCEPTIONAL CASES.

20MA CURRENT LOOP

FOUR LEADS ARE REQUIRED TO CONNECT THE BOARD TO A 20MA CURRENT LOOP DEVICE SUCH AS AN ASR33 TELETYPE:

CURRENT LOOP INPUT + AND -

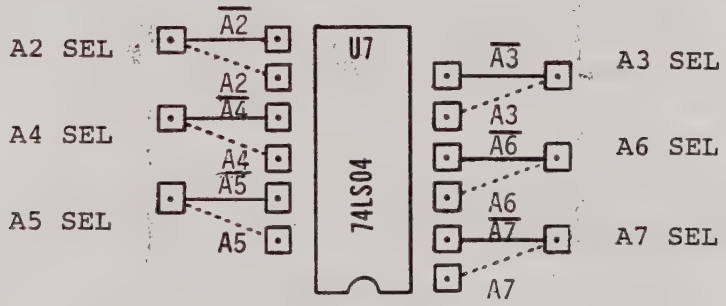
CURRENT LOOP OUTPUT + AND -

THESE SIGNALS ARE AVAILABLE ON THE PADS IMMEDIATELY ABOVE J3. REFER TO FIGURE 3-2 FOR JUMPERING ILLUSTRATION. A MOLEX CONNECTOR STRIP SOLDERED IN THESE PADS CAN BE USED FOR THE TTY CONNECTIONS, AND A JUMPER MUST BE INSTALLED IN THE UPPER RIGHT HAND CORNER OF THE BOARD TO CONNECT THE CURRENT LOOP INPUT CIRCUIT.

FOR THIS APPLICATION, THE EIA RECEIVER (1489) SHOULD REMAIN IN PLACE AS IT PROVIDES AN INTERNAL RESISTIVE PULL-UP FOR Q1. BOTH CURRENT LOOP AND RS232 TERMINALS CAN BE CONNECTED SIMULTANEOUSLY, FOR EXAMPLE, WHERE A VIDEO DISPLAY AND TELETYPE PRINTER ARE USED. THE KEYBOARD INPUTS WOULD BE IN PARALLEL DUE TO THE WIRED "OR" CONNECTION OF RXD AND BOTH OUTPUTS WOULD BE ENABLED. THE BAUD RATE COULD BE SELECTED WITH AN EXTERNAL SWITCH, AND THE TTY PRINTER COULD BE DISABLED WITH AN SPST SWITCH FROM CLO- TO GROUND.

CHANGING THE BOARD ADDRESS

ON EITHER SIDE OF U7 IN THE LOWER LEFT CORNER OF THE BOARD ARE SIX SETS OF PADS THAT CONTROL THE BOARD ADDRESS.



TO CHANGE THE ADDRESS, CUT THE TRACE BETWEEN PADS NEXT TO U7 AND THE ADDRESS BIT TO BE CHANGED AND INSTALL A JUMPER AS INDICATED BY THE DOTTED LINE TO THE INVERTED PIN ON U7. FOR EXAMPLE, TO CHANGE THE BOARD ADDRESS FROM PORT 0 TO PORT 4 CUT THE TRACE FROM A2 TO A2SEL AND ADD A JUMPER FROM A2 TO A2SEL.

EXTERNAL BAUD RATE CONTROL

IF TWO DIFFERENT BAUD RATES ARE COMMONLY USED, THE DIP SWITCH CAN BE REPLACED WITH AN SPDT SWITCH MOUNTED ON THE REAR PANEL AND CONNECTED TO THE BOARD WITH A 16 PIN DIP PLUG AND SOCKET. ALTERNATELY, THE BAUD RATE CAN BE CONTROLLED BY SOFTWARE BY SELECTING A BAUD RATE FACTOR OF 1X, 16X OR 64X WHEN THE USART

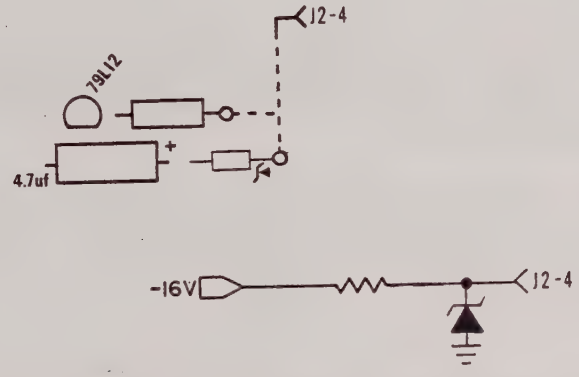
COMMAND IS ISSUED.

PATCH AREA

IMMEDIATELY TO THE RIGHT OF THE 8251 USART ARE LOCATED TWO 16 PIN PAD ARRAYS THAT CAN BE USED FOR ANY SPECIAL PURPOSE, SUCH AS CUSTOM I/O INTERFACES. THERE ARE SEVERAL UNUSED GATES AND INVERTERS ON THE BOARD WHICH HAVE 3 HOLE SOLDER PADS FOR SIMILAR PURPOSES. A BUFFERED CLK SIGNAL IS AVAILABLE AT U24-6.

KEYBOARD CONNECTION

ONE USE OF THE PARALLEL INPUT PORTS IS TO INTERFACE WITH A KEYBOARD. MANY KEYBOARDS REQUIRE A NEGATIVE SUPPLY VOLTAGE IN ADDITION TO THE +5V, AND THERE ARE PADS PROVIDED IN THE LOWER LEFT PORTION OF THE BOARD FOR AN OPTIONAL ZENER REGULATED SUPPLY.



THE ZENER VOLTAGE WILL DEPEND ON THE KEYBOARD REQUIREMENTS IF IT IS NEEDED AT ALL, AND THE RESISTOR SHOULD BE SELECTED TO BIAS THE ZENER WITH AT LEAST 10MA OF CURRENT IN ADDITION TO THE CURRENT REQUIRED BY THE KEYBOARD.

FOR EXAMPLE, WITH A KEYBOARD REQUIRING 10MA OF CURRENT AT 6V, THE ZENER COULD BE A 1N752A (5.6V) AND THE RESISTOR COULD BE 10/.02 = 500 OHMS (470 NOMINAL). THE ZENER DISSIPATION WOULD BE 60MW AND THE RESISTOR DISSIPATION WOULD BE 200MW (USE A 1/2 WATT RESISTOR TO ALLOW FOR HIGHER SUPPLY VOLTAGE).

THE DATA FROM THE KEYBOARD CAN BE CONNECTED TO INPUT PORT B, BITS 0-6. SOME KEYBOARDS HAVE POSITIVE TRUE DATA WHILE OTHERS HAVE NEGATIVE TRUE POLARITY. MOST SOFTWARE EXPECTS POSITIVE TRUE INPUTS, ALTHOUGH A CMA INSTRUCTION CAN BE USED TO INVERT THE INPUT DATA. ALTERNATELY, MANY KEYBOARDS HAVE INVERTING OUTPUT BUFFERS THAT CAN BE REPLACED WITH NON-INVERTING TYPES.

EITHER A KEYPRESS SIGNAL OR A STROBE PULSE IS GENERATED BY THE KEYBOARD TO INDICATE THAT A KEY HAS BEEN DEPRESSED AND THE DATA IS VALID (DEBOUNCED). THIS SIGNAL SHOULD BE CONNECTED TO STROBE INPUT B (J2 PIN 14). THE PROPER STROBE POLARITY SHOULD BE SELECTED BY INSTALLING A JUMPER BETWEEN U25 PIN 1 OR 2 AND THE PAD OPPOSITE THEM SO THAT THE SIGNAL APPLIED TO U18 PIN 9 GOES LOW WHEN A KEY IS PRESSED. THE Q OUTPUT OF U18 AVAILABLE AT J2 PIN 2 SHOULD BE CONNECTED TO J2 PIN 16 (A6 INPUT). THE KEYBOARD SHOULD OPERATE PROPERLY WITH THE

FOLLOWING INPUT ROUTINE:

```
INPUT   IN      0       KBD STATUS PORT A
        ANI     40H     STATUS BIT
        JZ      INPT
        IN      1       KBD DATA PORT B
        ANI     7FH     IGNORE EIGHTH BIT
        RET
```

ALTHOUGH OTHER CONFIGURATIONS ARE POSSIBLE, THE IMPORTANT THING TO REMEMBER IS THAT THE STATUS BIT LATCH U18 SHOULD NOT BE RESET UNTIL THE DATA IS READ IN FOR PROPER OPERATION WITH EXISTING SOFTWARE THAT MAY MAKE MULTIPLE TESTS OF THE STATUS BIT IN TESTING FOR A CONTROL C. THE STATUS BIT IS RESET WHEN PORT B IS INPUT. THE KEYBOARD DATA IS NOT LATCHED ON THE BOARD, AND THIS MEANS CONTROL C OR OTHER BREAK CHARACTER MUST BE HELD DOWN TO INTERRUPT PROGRAM EXECUTION UNTIL THE KEYBOARD IS POLLED. THERE ARE EXTREMELY RARE (FORTUNATELY) KEYBOARDS THAT DO NOT GENERATE STABLE OUTPUT DATA WHEN THE KEYS ARE DEPRESSED. DO NOT USE THIS TYPE OF KEYBOARD.

MOST KEYBOARDS HAVE EXTRA KEYS THAT ARE NOT CONNECTED TO THE ENCODER, BUT MAY BE USED INDEPENDENTLY. IT IS CONVENIENT TO CONNECT ONE OF THESE KEYS TO PIN 75 ON THE BUS (PRESET) TO PERMIT JUMPING TO THE MONITOR IN THE VECTOR 1 WITHOUT USING THE RESET SWITCH ON THE FRONT PANEL.

UNDERSHOOT ON THE CLOCK (PIN 49) BUS LINE CAN IN SOME CASES CAUSE INCORRECT BAUD RATES TO BE GENERATED. IF THIS OCCURS, TERMINATE THIS BUS LINE. A 220 OHM 1/4 WATT RESISTOR SOLDERED FROM THE EYELET ABOVE PIN 49 TO THE GROUND STRIP CONNECTED TO PIN 50 WILL CORRECT THE PROBLEM.

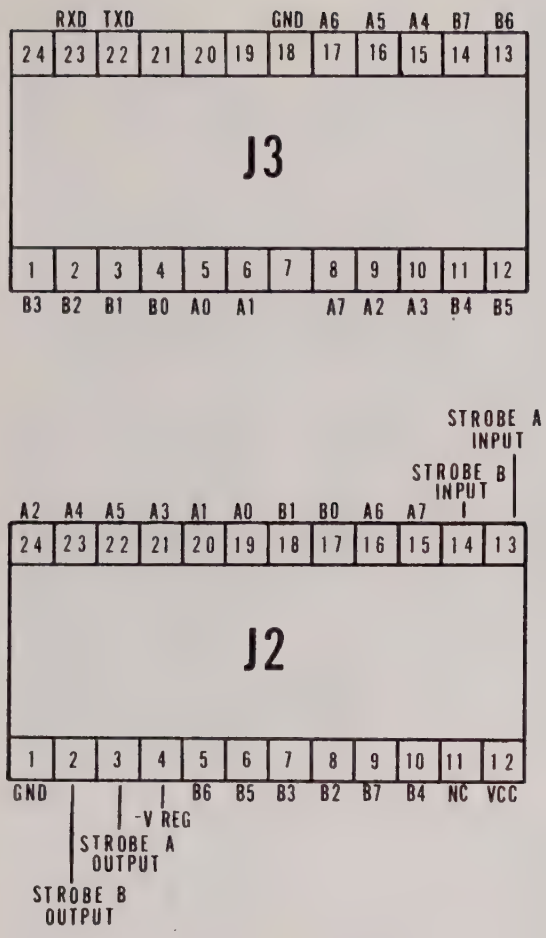


FIGURE 1

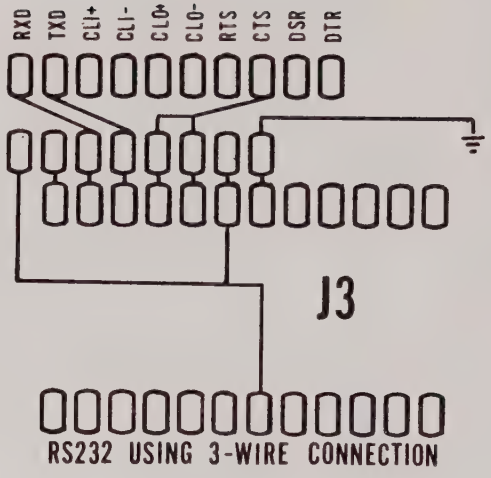

RXD TXD C1I+ C1I- C1O+ C1O- RTS CTS DSR DTR
J3
RS232 USING 3-WIRE CONNECTION
ON DB-25 - REAR PANEL CONNECT:
J3-23 to pin 2
J3-22 to pin 3
J3-18 to pin 7
pin 4 to pin 5
pin 20 to pin 6

FIGURE 2

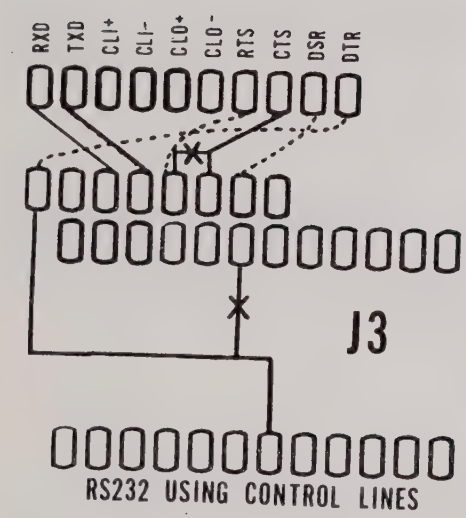
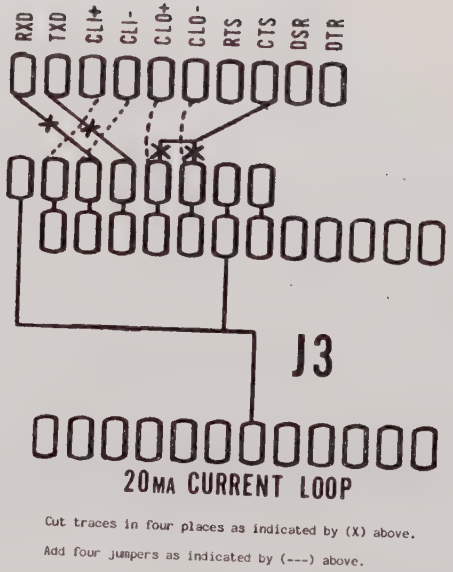


FIGURE 3-1



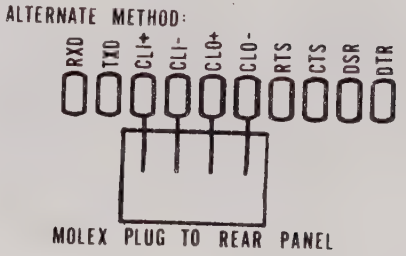
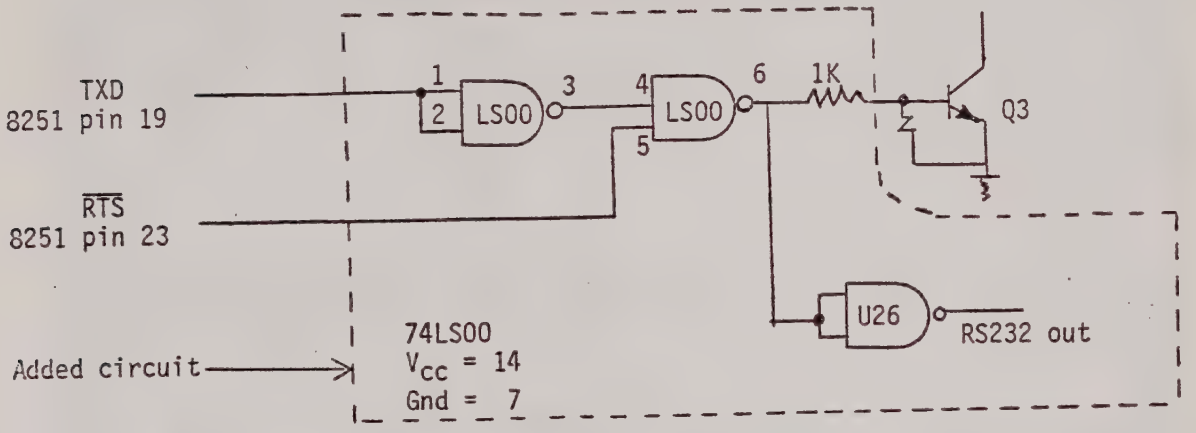


FIGURE 3-2

TTL LEVEL INPUT DEVICES SUCH AS THE TELETYPE MODEL 43 OPTION AAA (WHICH IS ABOUT $150 LESS EXPENSIVE THAN THE EQUIVALENT MODEL WITH EIA LEVEL DRIVERS) CAN BE EASILY ACCOMMODATED BY REPLACING THE EIA DRIVERS U26 AND U27 WITH JUMPERS. FOR EXAMPLE, JUMPER U27-6 TO U27-7 AND JUMPER U27-8 TO U27-10. JUMPER U26-5 TO U26-6.

MODIFICATION FOR TWO SERIAL OUTPUTS



1. REMOVE R5 CONNECTED TO THE BASE OF Q3.

2. ADD A 74LS00 IN THE KLUGE AREA OF THE BOARD.

3. WIRE UP THE CIRCUIT AS SHOWN USING THE RESISTOR REMOVED IN (1) AND AN UNUSED SECTION OF U26 (1488). A PRINTER CAN BE CONNECTED TO EITHER THE CURRENT LOOP OUTPUT OR THE RS232 OUTPUT AND IT WILL BE ENABLED BY CONTROL OF THE RTS BIT BY OUTPUTTING TO PORT 3. CARE SHOULD BE TAKEN THAT THE OTHER COMMAND BITS ARE OUTPUT CORRECTLY. SOFTWARE TO ENABLE AND DISABLE THE NEW OUTPUT WOULD BE AS FOLLOWS:

```
        MVI A, 27H
        OUT 3           SETS RTS LOW; DISABLES 2ND OUTPUT
-------------------------------------------------------------
        MVI A, 07H
        OUT 3           SETS RTS HIGH; ENABLES 2ND OUTPUT
```

THE MONITOR OPTION C INITIALIZES THE USART AND SETS RTS LOW, DISABLING THE SECOND OUTPUT. THE BAUD RATE CAN BE CHANGED BY DOING A SOFTWARE RESET OF THE USART AND REINITIALIZING THE USART FOR A DIFFERENT BAUD RATE FACTOR. FOR EXAMPLE, IF A VIDEO DISPLAY IS USED AT 1200 BAUD WITH THE NORMAL X16 BAUD RATE FACTOR, REINITIALIZING WITH X64 WOULD GIVE A 300 BAUD OUTPUT. IT WOULD BE NECESSARY TO USE THE REMAINING SECTIONS OF THE 74LS00 TO DISABLE THE VIDEO OUTPUT WHILE OUTPUTTING TO THE PRINTER AT THE LOWER SPEED. CARE WOULD ALSO HAVE TO BE EXERCISED TO PREVENT THE SOFTWARE RESET FROM DUMPING THE CHARACTER BEING TRANSMITTED BY THE USART. IT IS NECESSARY TO TEST THE TXE STATUS BIT TO ENSURE THIS.

GENERAL TROUBLE SHOOTING GUIDE

BECAUSE OF THE COMPLEXITY OF THE ENTIRE COMPUTER SYSTEM, BOTH HARDWARE AND SOFTWARE, IT IS ESSENTIAL TO ISOLATE ANY PROBLEM TO AN INDIVIDUAL BOARD OR PROGRAM. FORTUNATELY, ALL OF THE COMPUTER LOGIC IS ON EASILY REMOVABLE BOARDS. IT IS EXTREMELY VALUABLE TO HAVE ACCESS TO A TESTED COMPUTER SO THAT THE BOARDS CAN BE INDIVIDUALLY TESTED. ALTHOUGH THERE IS THE POSSIBILITY OF INTERACTION BETWEEN BOARDS DUE TO MARGINAL TIMING OR DEFECTIVE COMPONENTS, THIS IS NOT THE USUAL CASE, AND IT IS BEST TO ASSUME THAT IF A BOARD WORKS IN COMPUTER "A" IT WILL ALSO WORK IN COMPUTER "B".

THE MINIMUM SYSTEM CONSISTS OF THREE BOARDS: THE CPU BOARD, THE PROM/RAM BOARD, AND EITHER A VIDEO OR SERIAL I/O BOARD. MAKE SURE THAT THE MONITOR PROGRAM HAS BEEN PROPERLY PATCHED FOR THE PARTICULAR I/O CONFIGURATION OF YOUR SYSTEM. THERE IS TOTAL CONFUSION IN THE INDUSTRY CONCERNING PORT ASSIGNMENTS, LOGIC CONVENTIONS, AND STRAPPING OPTIONS. SEVERAL TYPES OF PROGRAMMABLE USARTS ARE USED WHICH REQUIRE INITIALIZATION.

IF YOU HAVE CAREFULLY FOLLOWED THE ASSEMBLY INSTRUCTIONS FOR EACH OF THE BOARDS AND THE REGULATORS CHECK OUT, INSTALL ALL CHIPS. LET'S ASSUME YOU ARE USING A VIDEO DISPLAY. AS SOON AS YOU TURN THE COMPUTER ON, YOU SHOULD SEE A DISPLAY OF RANDOM MEMORY GARBAGE ON THE TV SCREEN. THIS WILL BE INDEPENDENT OF ANY FUNCTIONING OF THE COMPUTER OTHER THAN THE CLOCK OSCILLATOR. IF YOU DO NOT GET A PROPER DISPLAY, THE VIDEO INTERFACE MUST BE DEBUGGED FIRST. FEEL THE CHIPS ON THE BOARD. ANY THAT ARE HOT TO THE TOUCH MAY BE IN BACKWARD (PROBABLY DESTROYED IF TTL) OR MAY HAVE THEIR OUTPUTS SHORTED. THERE IS MORE THAN A FACTOR OF TEN DIFFERENCE IN THE POWER DISSIPATION OF TTL CHIPS, BUT THEY SHOULD NOT BE UNCOMFORTABLY HOT TO THE TOUCH.

REMOVE THE BOARD AND INSPECT IT CAREFULLY. ABOUT HALF OF THE PROBLEMS CAN BE FOUND SIMPLY BY VISUAL INSPECTION. LOOK WITH A MAGNIFYING GLASS OR INSPECTION SCOPE AT EACH PIN ON THE BOTTOM FOR UNSOLDERED PINS, MISSING PINS THAT MAY BE BENT UNDER OR BROKEN OFF, SOLDER BRIDGES BETWEEN PINS OR TO ADJACENT TRACES, AND ETCH BRIDGES BETWEEN TRACES (VERY HARD TO SEE). A CAREFUL EXAMINATION WILL TAKE 15 MINUTES, BUT MAY SAVE YOU A LOT OF GRIEF, AND YOU MAY DISCOVER PROBLEMS LIKE UNSOLDERED PINS THAT MAY REVEAL THEMSELVES ONLY LATER AS INTERMITTENT PROBLEMS. EXAMINE THE TOP OF THE BOARD TO BE SURE THE PROPER CHIPS ARE INSTALLED IN THE RIGHT PLACES. SIGHT ALONG THE EDGE OF THE CHIPS TO FIND BENT UNDER PINS. CHIPS ARE SOMETIMES INSERTED WITH A WHOLE ROW OF PINS THAT MISS THE SOCKET HOLES.

IF THE VISUAL INSPECTION FAILS TO GET THE VIDEO DISPLAY WORKING, A COMPONENT MAY BE BAD (USUALLY AN IC). TRY EXCHANGING IDENTICAL COMPONENTS TO SEE IF THE SYMPTOMS CHANGE. AT THIS POINT IT IS WISE TO GO BACK AND CAREFULLY REREAD THE MANUAL TO BE SURE YOU UNDERSTAND THE WAY THE BOARD WORKS AND THAT YOU HAVE SELECTED THE PROPER JUMPER OPTIONS. AFTER THIS, YOU WILL PROBABLY WANT TO TAKE THE UNIT TO A DEALER IF YOU ARE NOT FAMILIAR WITH DIGITAL TROUBLE SHOOTING PROCEDURES, OR GO THROUGH THE CIRCUIT BLOCK BY BLOCK WITH A SCOPE OR LOGIC PROBE IF YOU ARE EXPERIENCED.

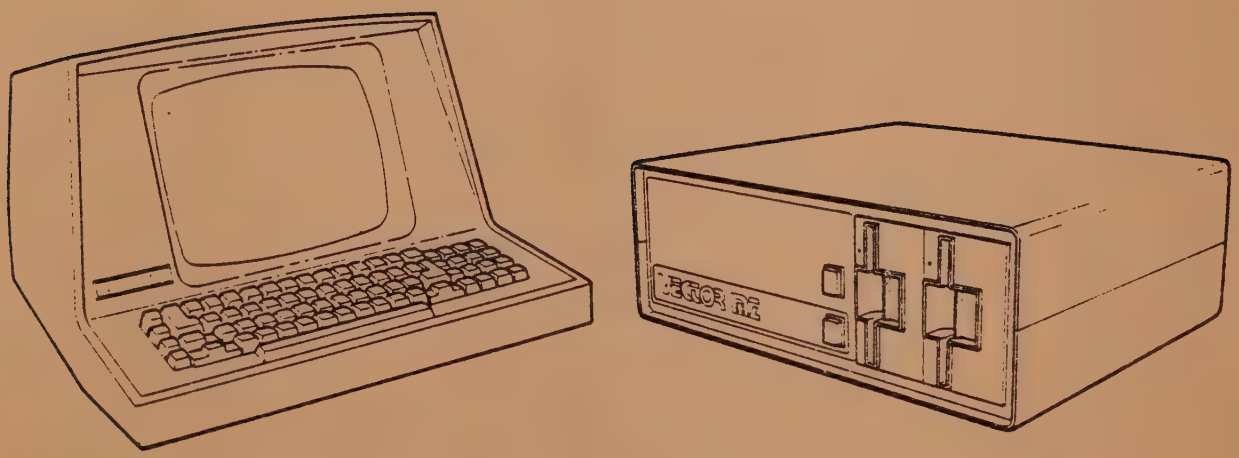
AFTER THE VIDEO DISPLAY OR SERIAL I/O IS WORKING, THE RESET SWITCH SHOULD CAUSE A "*" PROMPT TO BE WRITTEN. IF THIS DOES NOT WORK, FOLLOW THE SAME PROCEDURE ON

THE CPU AND PROM/RAM BOARDS. THE CPU BOARD CONSISTS MOSTLY OF 8097 BUS DRIVERS WHICH CAN BE EXCHANGED ONE BY ONE. THE VECTORED INTERRUPT AND REAL TIME CLOCK COMPONENTS, IC A1, ARE NOT NECESSARY IN THE BOARD AT THIS TIME AND SHOULD BE REMOVED. USING A SCOPE, EXAMINE THE OUTPUT PINS OF ALL CHIPS. LOW LOGIC LEVELS ARE NORMALLY LESS THAN 0.2 VOLTS AND HIGH GREATER THAN 3.0 VOLTS. A LEVEL OF 0.4 VOLTS MAY INDICATE SHORTS BETWEEN OUTPUTS WHERE ONE IS TRYING TO PULL HIGH AND THE OTHER LOW. A LEVEL OF 1.2 VOLTS INDICATES AN OPEN CIRCUITED INPUT. NMOS CHIPS HAVE SIMILAR LOGIC LEVELS, WHILE PMOS CHIPS CAN PULL TTL INPUTS TO -0.6V WHERE THE INPUT CLAMP DIODE LIMITS THE VOLTAGE. DO NOT BE SURPRISED AT HOW STRANGE SOME OF THE WAVEFORMS ON THE BUS LOOK, SUCH AS THE DI LINES. THERE ARE PERIODS OF TIME DURING WHICH THE BUS IS NOT BEING ACTIVELY DRIVEN, AND THE VOLTAGE MAY DRIFT DUE TO RECEIVER INPUT CURRENT. ABNORMAL OPERATION IS INDICATED PRINCIPALLY BY ABNORMAL LOGIC LEVELS MAINTAINED CONSTANT FOR AT LEAST ONE CLOCK PERIOD (500 MICROSECONDS).

ONCE YOUR BASIC SYSTEM IS WORKING, CHECK OUT OF MEMORY BOARDS AND OTHER INTERFACES IS RELATIVELY STRAIGHTFORWARD USING THE MEMORY TEST PROGRAM IN THE MONITOR, OR SIMPLE DIAGNOSTIC ROUTINES YOU CAN PROGRAM IN MEMORY ON THE PROM/RAM BOARD. AFTER YOUR SYSTEM IS UP AND RUNNING, IT SHOULD BE QUITE RELIABLE. SINCE MOST MICROCOMPUTER SYSTEMS ARE MEMORY INTENSIVE, THE MEMORY IS THE MOST LIKELY SOURCE OF COMPONENT FAILURE. A SYSTEM WITH 32K OF STATIC MEMORY MAY CONTAIN 75% OF ITS COMPONETS ON THE MEMORY BOARDS. IF A PROBLEM IS EXPERIENCED RUNNING A PROGRAM, FIRST SUSPECT THE MEMORY AND USE THE MONITOR TEST PROGRAM. WE HAVE YET TO EXPERIENCE A PROBLEM WITH OUR 8K MEMORY BOARDS THAT WAS NOT REVEALED BY THE TEST PROGRAM. IF YOU DO MUCH REARRANGING OF YOUR SYSTEM, IT IS A GOOD PRACTICE TO TEST MEMORY FOR A FEW SECONDS WHEN YOU FIRST TURN ON THE COMPUTER TO MAKE SURE THE BOARDS ARE ADDRESSED PROPERLY OR THAT THEY ARE IN THE COMPUTER. THIS MAY SAVE SOME HEAD SCRATCHING WHEN THE PROGRAM YOU HAVE JUST LOADED FAILS TO RESPOND TO YOUR EAGER KEYBOARD TOUCH. IF YOU SUSPECT TEMPERATURE SENSITIVE CHIPS, REMOVE THE COVER OF THE COMPUTER TO INTERRUPT AIR FLOW BETWEEN BOARDS. WE DO NOT RECOMMEND OBSTRUCTING THE AIR FLOW THROUGH THE COMPUTER BY PLACING A SHEET OF PAPER OVER THE LEFT SIDE. A FULL COMPUTER MAY DISSIPATE OVER 300 WATTS AND REACH UNACCEPTABLE TEMPERATURES IF NO AIRFLOW IS PERMITTED.

PRECISION ANALOG INTERFACE BOARD

USERS MANUAL



VECTOR GRAPHIC INC.

PRECISION ANALOG INTERFACE BOARD

TABLE OF CONTENTS

SUBJECT	PAGE
INTRODUCTION	1
PARTS LIST	2
SCHEMATIC	3A
COMPONENT DIAGRAM	3B
THEORY OF OPERATION	4
USERS GUIDE	6
GENERAL TROUBLE SHOOTING GUIDE	9

INTRODUCTION

THE VECTOR GRAPHIC PRECISION ANALOG INTERFACE BOARD (PAIB) ALLOWS YOUR S-100 BUS COMPUTER TO INPUT OR OUTPUT ANALOG SIGNALS BY USE OF HIGH ACCURACY 12-BIT ANALOG TO DIGITAL AND DIGITAL TO ANALOG CONVERTERS. ITS INTERFACE CAPABILITIES ADD A NEW DIMENSION TO DIGITAL SYSTEMS. THIS USERS GUIDE WILL PROVIDE YOU WITH THEORY OF OPERATION AND APPLICATIONS INFORMATION.

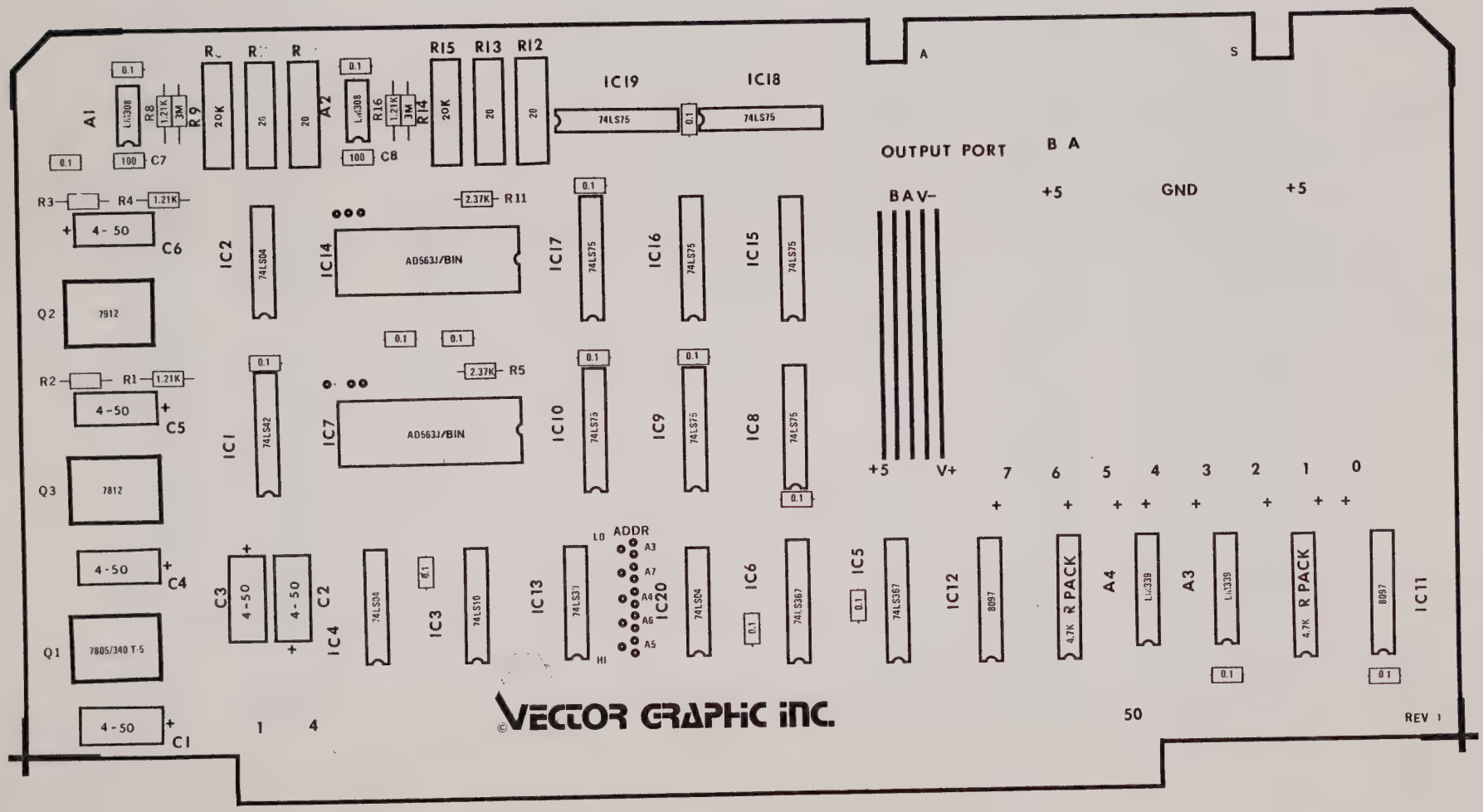
IF THERE IS ANYTHING THAT YOU DO NOT UNDERSTAND, PLEASE DO NOT HESITATE TO CALL OR WRITE US!

PLEASE FILL OUT AND RETURN THE WARRANTY CARD SO THAT WE CAN ADD YOU TO OUR MAILING LIST FOR FUTURE PRODUCTS.

PARTS LISTS

1	PRINTED CIRCUIT BOARD
1	7805/340T-5 REGULATOR (Q1)
1	7812 REGULATOR (Q3)
1	7912 REGULATOR (Q2)
2	74367/8097 (IC11, IC12)
3	74LS04 (IC2, IC4, IC20)
8	74LS75 (IC8-10, IC15-19)
1	74LS42 (IC1)
1	74LS10 (IC3)
1	74LS30 (IC13)
2	74LS367 (IC5, IC6)
2	LM308N (A1, A2)
2	LM339AN (A3, A4)
2	12 BIT D/A CONVERTERS (AD563J/BIN) (IC7, IC14)
16	.1 MFD 25V CAPACITOR
2	100 PF 50V CAPACITOR
6	4 MFD 50V AXIAL ELECTROLYTIC CAPACITOR
4	20 OHM 1/4 WATT TRIMPOT RESISTOR
2	20K OHM 1/4 WATT TRIMPOT RESISTOR
2	2.37K 1/2 WATT RESISTOR
4	1.2K 1/2 WATT RESISTOR
2	150 OHM 1/2 WATT CARBON RESISTOR
2	3M 1/2 WATT CARBON RESISTOR
2	4.7K 16 PIN DIP RESISTOR NETWORKS

QTY	DESCRIPTION
2	8 PIN SOCKETS
7	14 PIN SOCKETS
13	16 PIN SOCKETS
2	24 PIN SOCKETS
1	15 POSITION, DUAL READOUT EDGE CONNECTOR
1	MISCELLANEOUS HARDWARE, INSULATING WASHERS

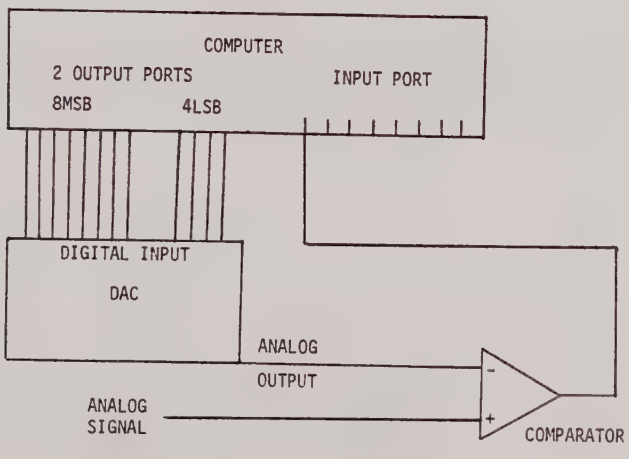

A1
A2
R9
R14
R8
R16
R15
R13
R12
IC19
IC18
74LS75
OUTPUT PORT
B A
B A V-
+5
GND
+5
R3
R4
1.21K
C6
Q2
7912
IC2
74LS04
IC14
AD563J/BIN
2.37K
R11
IC17
IC16
IC15
R2
R1
C5
IC1
74LS42
IC7
R5
IC10
IC9
IC8
Q3
7812
+5
V+
7
6
5
4
3
2
1
0
C4
LO
ADDR
HI
A3
A7
A4
A6
A5
IC20
IC6
IC5
IC12
8097
4.7K R PACK
A4
A3
IC11
C3
C2
IC4
IC3
IC13
Q1
7805/340 T-5
C1
VECTOR GRAPHIC INC.
50
REV 1

THEORY OF OPERATION

THE HEART OF THE VECTOR GRAPHIC PAIB CONSISTS OF TWO PRECISION DIGITAL TO ANALOG CONVERTERS, IC7 AND IC14. THESE INTEGRATED CIRCUITS ARE COMPOSED OF R-2R RESISTOR LADDER ARRAYS WITH BRANCHES SWITCHED BY ANALOG SWITCHES CONTROLLED BY AN INCOMING DIGITAL WORD. THE RESISTOR NETWORK WEIGHTS THE CURRENT CONTROLLED IN EACH BRANCH IN A BINARY FASHION, THEREFORE THE VOLTAGE PRODUCED AT THE OUTPUT IS PROPORTIONAL TO THE DIGITAL WORD. A TWELVE-BIT DAC ACCEPTS DIGITAL INPUTS FROM 000H TO FFFH AND PRODUCES AN ANALOG OUTPUT THAT VARIES OVER 4096 DISCRETE LEVELS.

WHEN AN OUTPUT INSTRUCTION ADDRESSES A VALUE WHOSE 5 MSB MATCH THOSE JUMPERED IN THE ADDRESS SELECTION AREA, INVERTERS IC20 AND NAND GATE IC13 ENABLE THE DECODER IC1 TO LOOK AT THE THREE LSB OF THE ADDRESS AND SELECT WHICH OF THE QUAD LATCHES IC8-10, IC15-18 LATCH DATA OFF OF THE BUS. THE QUAD LATCH OUTPUTS CONNECT DIRECTLY TO THE DAC.

THE DAC ALSO PLAYS AN IMPORTANT ROLE IN THE COMPLEMENTARY FUNCTON OF ANALOG TO DIGITAL CONVERSION. THE METHOD USED FOR A TO D CONVERSION IS CALLED SUCCESSIVE APPROXIMATION. A SKETCH OF THE IMPORTANT ELEMENTS IN A SUCCESSIVE APPROXIMATION A TO D CONVERTER (ADC) ARE SHOWN BELOW.



ONE DAC OUTPUT IS CONNECTED TO THE INVERTING INPUT OF A COMPARATOR AND THE ANALOG INPUT TO BE CONVERTED IS CONNECTED TO THE NON-INVERTING INPUT. THE COMPARATOR'S OUTPUT CONNECTS TO AN INPUT PORT SO THAT ITS STATE MAY BE MONITORED BY THE COMPUTER.

WHEN THE ADC ROUTINE STARTS, THE COMPUTER OUTPUTS A 12-BIT DIGITAL WORD TO THE DAC WITH ONLY THE MSB HIGH (800H) AND THIS CAUSES THE DAC OUTPUT TO GO TO HALF OF ITS OUTPUT RANGE. THE COMPUTER THEN IS GIVEN AN IN COMMAND WHICH ALLOWS IT TO EXAMINE THE COMPARATOR'S OUTPUT. IN THIS MANNER, THE COMPUTER DETERMINES

WHETHER THE ANALOG INPUT IS GREATER THAN OR LESS THAN HALF OF THE OUTPUT RANGE. IF THE ANALOG INPUT IS LESS THAN THE DAC OUTPUT, THE MSB IS RESET TO 0. OTHERWISE, THE MSB IS LEFT AS 1. THE NEXT STEP OF THE CONVERSION ROUTINE OUTPUTS ANOTHER 12-BIT SIGNAL TO THE DAC WITH THE SECOND MSB HIGH AND THE MSB FIXED AS DETERMINED IN THE FIRST STEP. THE COMPARATOR OUTPUT IS THEN EXAMINED AGAIN. THIS PROCEDURE IS CONTINUED UNTIL ALL TWELVE BITS ARE TESTED IN SUCCESSION. AT THE CONCLUSION OF THE ROUTINE THE COMPUTER CONTAINS THE CONVERTED DIGITAL WORD. DETAILED ASSEMBLY LANGUAGE LISTINGS OF SUCCESSIVE APPROXIMATION ROUTINES ARE INCLUDED IN THE USERS GUIDE.

YOUR PAIB HAS EIGHT COMPARATORS THUS ENABLING UP TO EIGHT ANALOG SIGNALS TO BE CONVERTED TO DIGITAL VALUES. BECAUSE A WIDE RANGE OF ANALOG OUTPUT SIGNALS MAY BE DESIRED, OR A WIDE RANGE OF ANALOG INPUTS MAY HAVE TO BE CONNECTED, A PATCH AREA ON THE BOARD IS PROVIDED FOR YOUR CONVENIENCE.

AS AN EXAMPLE OF THE USE OF THE PATCH AREA CONSIDER THE APPLICATION OF THE PAIB TO OUTPUT A -5 TO +11 VOLT SIGNAL TO SOME TEST DEVICE AND INPUT VOLTAGES FROM THREE THERMOCOUPLES. DAC #A WILL BE USED FOR OUTPUT AND DAC #B WILL BE USED FOR THE ADC.

DAC #A SHOULD BE CONFIGURED FOR 0-10 VOLT UNIPOLAR OPERATION AND THE OUTPUT SHOULD BE CONNECTED TO A X1.5 OP-AMP CIRCUIT WITH OFFSET IN THE PATCH AREA. THE OUTPUT OF THE AMPLIFIER MAY BE BROUGHT OUT TO ANY FREE PIN ON THE EDGE CONNECTOR.

FOR THE ADC CONNECT THE OUTPUT OF DAC #B TO THE INVERTING INPUTS OF COMPARATORS 1,2 AND 3. THE THERMOCOUPLE SIGNALS MAY HAVE TO BE AMPLIFIED USING INSTRUMENTATION AMPLIFIERS IN THE PATCH AREA. EACH AMPLIFIER OUTPUT SHOULD CONNECT TO ONE OF THE NON-INVERTING INPUTS TO COMPARATOR 1, 2, AND 3.

USERS GUIDE

BOARD ADDRESS

THE PAIB INTERFACES TO THE COMPUTER BUS VIA ONE INPUT PORT AND FIVE OUTPUT PORTS. THE FIVE MSB OF THE BOARD ADDRESS ARE SELECTED BY PATCHING THE APPROPRIATE JUMPERS IN THE ADDRESS JUMPER AREA. THE PAIB IS PRE-JUMPERED FOR ADDRESS 10H. THE PORT ASSIGNMENTS ARE AS FOLLOWS

OUTPUT PORTS

10H	8 MSB OF DAC #A OUTPUT	
11H	4 LSB OF DAC #A OUTPUT	4 MSB OF BYTE UNUSED
12H	8 MSB OF DAC #B OUTPUT	
13H	4 LSB OF DAC #B OUTPUT	4 MSB OF BYTE UNUSED
14H	8-BIT CONTROL PORT.	

INPUT PORT

10H	CONNECTED TO COMPARATOR OUTPUTS

OUTPUTTING RANGE

THE DAC'S MAY BE CONFIGURED FOR EITHER UNIPOLAR OR BIPOLAR OUTPUT RANGES WITH A VARIETY OF VOLTAGE RANGES. THE OPTIONS ARE AS FOLLOWS:

DAC OUTPUT RANGE		JUMPER (O=OP AMP OUTPUT)
0-10	UNIPOLAR	O TO 10
0-5	UNIPOLAR	O TO 10, SHORT 9 TO 11
+10	BIPOLAR	O TO 11
+5	BIPOLAR	O TO 10
+2.5	BIPOLAR	O TO 10, SHORT 9 TO 11

THE PAIB IS PRE JUMPERED FOR -5 VOLT TO + 5 VOLTS BIPOLAR OPERATION. FOR UNIPOLAR OPERATION ON CHANNEL A RSTET SHOULD BE REMOVED. R13 SHOULD BE REMOVED FOR UNIPOLAR OPERATION ON CHANNEL B.

ANALOG TO DIGITAL CONVERSION

THE PAIB IS SUPPLIED WITHOUT ANY CONNECTION FOR ANALOG TO DIGITAL CONVERSION AND ANY DESIRED PATCHES MUST BE ADDED BY THE USER. EITHER CHANNEL A OR CHANNEL B OR BOTH MAY BE USED AS ADC'S. TO USE CHANNEL B AS AN ADC FOR THREE INCOMING SIGNALS BETWEEN -5 AND +5 VOLTS, FOR EXAMPLE, CONNECT THE "B" BUS NEAR THE PATCH AREA TO THREE OF THE INVERTING INPUTS OF THE COMPARATOR. (IC11, IC12). THE CHANNEL B DAC IS CONFIGURED FOR -5 VOLT TO +5 VOLT BIPOLAR OPERATION AS SUPPLIED. CONVERT EACH OF THE THREE INCOMING ANALOG SIGNALS TO A NON-INVERTING INPUT OF THE COMPARATORS. THIS MAY BE EASILY DONE BY CONNECTING THE APPROPRIATE ANALOG INPUT PAD NEAR THE 15 POSITION EDGE CONNECTOR TO A CORRESPONDING "+" PAD NEAR THE COMPARATORS.

ANALOG INPUT PAD NEAR THE 15 POSITION EDGE CONNECTOR TO A CORRESPONDING "+" PAD NEAR THE COMPARATORS.

THE PAIB IS NOW PATCHED FOR ADC CONVERSION AND RUNNING THE ADC ROUTINES OUTLINED IN THE APPENDIX, AFTER SELECTING THE APPROPRIATE CHANNELS, WILL ALLOW SUCCESSFUL ANALOG TO DIGITAL CONVERSION.

EDGE CONNECTOR PIN ASSIGNMENTS

ALL ANALOG INPUT AND OUTPUT SIGNALS AND THE EIGHT CONTROL LINES INTERFACE TO THE PAIB VIA A 15 POSITION, DUAL READOUT EDGE CONNECTOR. ITS PIN ASSIGNMENTS ARE AS FOLLOWS:

A CONTROL 2	1 CONTROL 3	
B CONTROL 1	2 NC	(NC MEANS NO CONNECTION)
C CONTROL 0	3 NC	
D CONTROL 7	4 NC	
E CONTROL 6	5 NC	
F CONTROL 4	6 CONTROL 5	
H DAC #B OUT	7 NC	
J DAC #A OUT	8 ANALOG IN 7	
K NC	9 ANALOG IN 6	
L NC	10 ANALOG IN 5	
M NC	11 ANALOG IN 4	
N NC	12 ANALOG IN 3	
P NC	13 ANALOG IN 2	
R NC	14 ANALOG IN 1	
S GROUND	15 ANALOG IN 0	

BASIC INTERFACE

THE MACHINE LANGUAGE ADC ROUTINE MAY BE EASILY CALLED AS A MACHINE LANGUAGE ROUTINE IN A BASIC PROGRAM. CONVERTED VALUES MAY BE RETURNED TO THE BASIC USING THE PEEK COMMAND.

THE FOLLOWING IS A SHORT BASIC PROGRAM TO OUTPUT A VOLTAGE FROM DAC #A AND USE DAC #B AS AN ADC TO INPUT THE VALUE. THE PAIB IS ADDRESSED AT 10H AND BOTH CONVERTERS ARE CONFIGURED FOR 0-10 VOLT OPERATION

```
10  PRINT"DAC-ADC TEST PROGRAM"
20  INPUT"OUTPUT VOLTAGE";V
30  V=V*409.5
40  OUT 16, V/16
50  OUT 17, V AND 15
60  REM CALL MACHINE LANGUAGE ADC ROUTINE HERE
70  REM EXAMPLE IN VECTOR BASIC. INSERT X,Y,M APPROPRIATE FOR YOUR SYSTEM.
80  POKE 10834,X
90  POKE 10835,Y 100 V=USR(1)
110 V1=(16*PEEK(M)+(PEEK(M+1)AND 15))/4095
120 PRINT"INPUT VOLTAGE IS";V1
130 GO TO 20
```

ADC SOFTWARE

TWO ASSEMBLY LANGUAGE PROGRAM LISTINGS FOLLOW. THE FIRST IS A SIMPLE, ONE CHANNEL ANALOG TO DIGITAL CONVERSION ROUTINE. THE SECOND IS A MORE ELABORATE THREE CHANNEL CONVERSION ROUTINE WITH A SIXTEEN TIMES SUMMING OF THE CONVERTED VALUE FOR SMOOTHING OF THE DATA.

NOISE

TO TAKE FULL ADVANTAGE OF THE PRECISION OF THE PAIB IT MUST BE RECOGNIZED THAT THE COMPUTER IS AN EXTREMELY NOISY ENVIRONMENT, AND PRECAUTIONS MUST BE TAKEN TO DISCRIMINATE AGAINST THIS NOISE. THREE PREDOMINANT FREQUENCY RANGES OF NOISE EXIST:

1) DC VOLTAGE OFFSETS MAY EXIST BETWEEN THE PAIB AND OTHER PARTS OF THE CHASSIS OR EXTERNAL EQUIPMENT OF THE MAGNITUDE OF 0.1V. DIFFERENTIAL AMPLIFIERS ARE AN EFFECTIVE WAY OF ELIMINATING THIS OFFET.

2) LINE FREQUENCY COMPONENTS AND HARMONICS CAN ALSO BE SUPPRESSED BY USING DIFFERENTIAL INPUTS. THE MAGNITUDE IS COMPARABLE TO 1).

3) HIGH FREQUENCY (>10 KHZ) NOISE IS CAUSED BY THE DIGITAL CIRCUITRY IN THE COMPUTER. THIS IS ABOVE THE FREQUENCY THAT CAN BE MEASURED BY THE PAIB AND SHOULD BE ELIMINATED BY LOW PASS FILTERS. AN EXAMPLE OF CIRCUITRY THAT CAN BE USED TO TRANSMIT SIGNALS BETWEEN TWO REMOTELY LOCATED PAIB BOARDS (OR OTHER EQUIPMENT) IS SHOWN IN FIGURE 1.

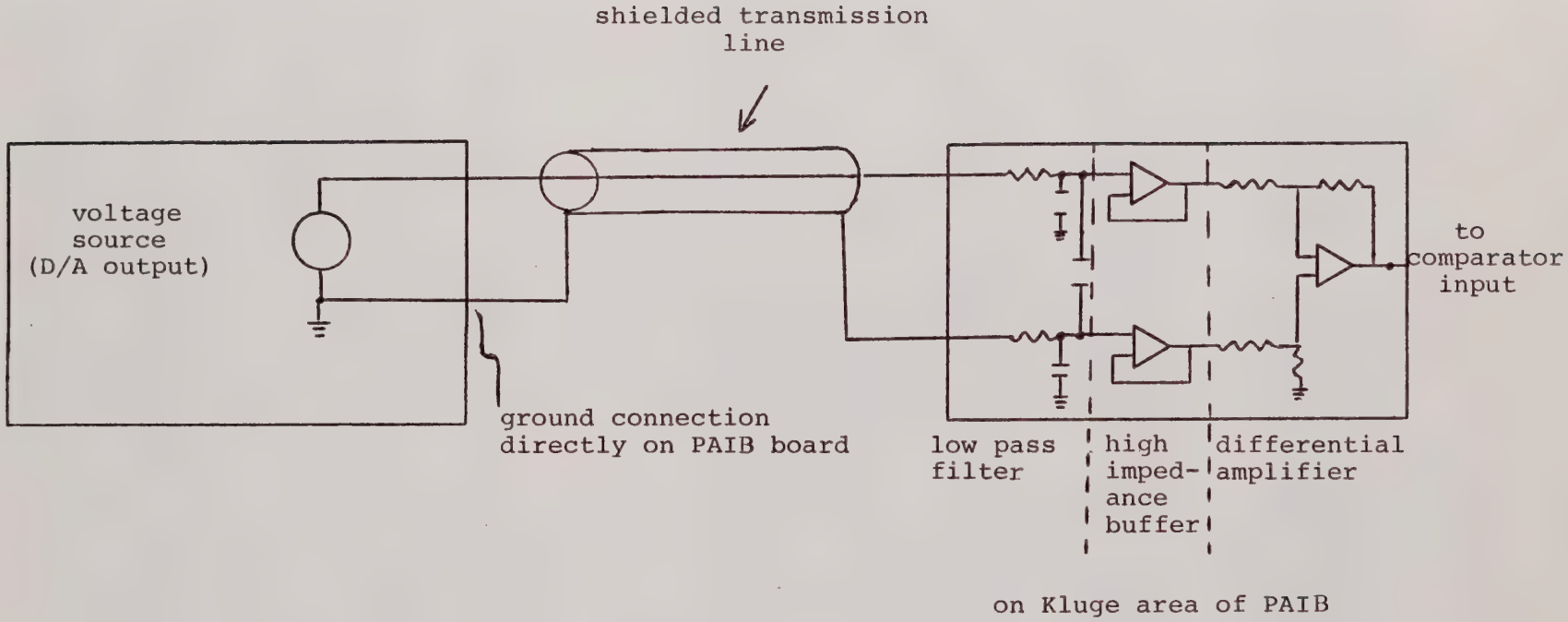

shielded transmission line
voltage source (D/A output)
ground connection directly on PAIB board
low pass filter
high imped-ance buffer
differential amplifier
to comparator input
on Kluge area of PAIB

Figure 1

PRECISION ANALOG INTERFACE BOARD

SPECIFICATIONS

INTERFACE	S-100 BUS COMPATIBLE ANALOG INPUTS AND OUTPUTS AVAILABLE AT 15-PIN CARD EDGE CONNECTOR.
OUTPUT CHANNELS	2
INPUT CHANNELS	8
RESOLUTION	1 PART IN 4096
CONVERSION TIME	360 USEC FOR FULL 12-BIT CONVERSION
CONTROL PORTS	1 INPUT PORT 5 OUTPUT PORTS
POWER REQUIREMENTS	+8V @ +16V @ -16V @
DAC	ANALOG DEVICES AD563J/BIN CONVERTER
DAC SETTING TIME	1.2 USEC

GENERAL TROUBLE SHOOTING GUIDE

BECAUSE OF THE COMPLEXITY OF THE ENTIRE COMPUTER SYSTEM, BOTH HARDWARE AND SOFTWARE, IT IS ESSENTIAL TO ISOLATE ANY PROBLEM TO AN INDIVIDUAL BOARD OR PROGRAM. FORTUNATELY, ALL OF THE COMPUTER LOGIC IS ON EASILY REMOVABLE BOARDS. IT IS EXTREMELY VALUABLE TO HAVE ACCESS TO A TESTED COMPUTER SO THAT THE BOARDS CAN BE INDIVIDUALLY TESTED. ALTHOUGH THERE IS THE POSSIBILITY OF INTERACTION BETWEEN BOARDS DUE TO MARGINAL TIMING OR DEFECTIVE COMPONENTS, THIS IS NOT THE USUAL CASE, AND IT IS BEST TO ASSUME THAT IF A BOARD WORKS IN COMPUTER "A" IT WILL ALSO WORK IN COMPUTER "B".

THE MINIMUM SYSTEM CONSISTS OF THREE BOARDS: THE CPU BOARD, THE PROM/RAM BOARD, AND EITHER A VIDEO OR SERIAL I/O BOARD. MAKE SURE THAT THE MONITOR PROGRAM HAS BEEN PROPERLY PATCHED FOR THE PARTICULAR I/O CONFIGURATION OF YOUR SYSTEM. THERE IS TOTAL CONFUSION IN THE INDUSTRY CONCERNING PORT ASSIGNMENTS, LOGIC CONVENTIONS, AND STRAPPING OPTIONS. SEVERAL TYPES OF PROGRAMMABLE USARTS ARE USED WHICH REQUIRE INITIALIZATION.

IF YOU HAVE CAREFULLY FOLLOWED THE ASSEMBLY INSTRUCTIONS FOR EACH OF THE BOARDS AND THE REGULATORS CHECK OUT, INSTALL ALL CHIPS. LET'S ASSUME YOU ARE USING A VIDEO DISPLAY. AS SOON AS YOU TURN THE COMPUTER ON, YOU SHOULD SEE A DISPLAY OF RANDOM MEMORY GARBAGE ON THE TV SCREEN. THIS WILL BE INDEPENDENT OF ANY FUNCTIONING OF THE COMPUTER OTHER THAN THE CLOCK OSCILLATOR. IF YOU DO NOT GET A PROPER DISPLAY, THE VIDEO INTERFACE MUST BE DEBUGGED FIRST. FEEL THE CHIPS ON THE BOARD. ANY THAT ARE HOT TO THE TOUCH MAY BE IN BACKWARD (PROBABLY DESTROYED IF TTL) OR MAY HAVE THEIR OUTPUTS SHORTED. THERE IS MORE THAN A FACTOR OF TEN DIFFERENCE IN THE POWER DISSIPATION OF TTL CHIPS, BUT THEY SHOULD NOT BE UNCOMFORTABLY HOT TO THE TOUCH.

REMOVE THE BOARD AND INSPECT IT CAREFULLY. ABOUT HALF OF THE PROBLEMS CAN BE FOUND SIMPLY BY VISUAL INSPECTION. LOOK WITH A MAGNIFYING GLASS OR INSPECTION SCOPE AT EACH PIN ON THE BOTTOM FOR UNSOLDERED PINS, MISSING PINS THAT MAY BE BENT UNDER OR BROKEN OFF, SOLDER BRIDGES BETWEEN PINS OR TO ADJACENT TRACES, AND ETCH BRIDGES BETWEEN TRACES (VERY HARD TO SEE). A CAREFUL EXAMINATION WILL TAKE 15 MINUTES, BUT MAY SAVE YOU A LOT OF GRIEF, AND YOU MAY DISCOVER PROBLEMS LIKE UNSOLDERED PINS THAT MAY REVEAL THEMSELVES ONLY LATER AS INTERMITTENT PROBLEMS. EXAMINE THE TOP OF THE BOARD TO BE SURE THE PROPER CHIPS ARE INSTALLED IN THE RIGHT PLACES. SIGHT ALONG THE EDGE OF THE CHIPS TO FIND BENT UNDER PINS. CHIPS ARE SOMETIMES INSERTED WITH A WHOLE ROW OF PINS THAT MISS THE SOCKET HOLES.

IF THE VISUAL INSPECTION FAILS TO GET THE VIDEO DISPLAY WORKING, A COMPONENT MAY BE BAD (USUALLY AN IC). TRY EXCHANGING IDENTICAL COMPONENTS TO SEE IF THE SYMPTOMS CHANGE. AT THIS POINT IT IS WISE TO GO BACK AND CAREFULLY REREAD THE MANUAL TO BE SURE YOU UNDERSTAND THE WAY THE BOARD WORKS AND THAT YOU HAVE SELECTED THE PROPER JUMPER OPTIONS. AFTER THIS, YOU WILL PROBABLY WANT TO TAKE THE UNIT TO A DEALER IF YOU ARE NOT FAMILIAR WITH DIGITAL TROUBLE SHOOTING PROCEDURES, OR GO THROUGH THE CIRCUIT BLOCK BY BLOCK WITH A SCOPE OR LOGIC PROBE IF YOU ARE EXPERIENCED.

AFTER THE VIDEO DISPLAY OR SERIAL I/O IS WORKING, THE RESET SWITCH SHOULD CAUSE A "*" PROMPT TO BE WRITTEN. IF THIS DOES NOT WORK, FOLLOW THE SAME PROCEDURE ON THE CPU AND PROM/RAM BOARDS. THE CPU BOARD CONSISTS MOSTLY OF 8097 BUS DRIVERS WHICH CAN BE EXCHANGED ONE BY ONE. THE VECTORED INTERRUPT AND REAL TIME CLOCK

COMPONENTS, IC A1, ARE NOT NECESSARY IN THE BOARD AT THIS TIME AND SHOULD BE REMOVED. USING A SCOPE, EXAMINE THE OUTPUT PINS OF ALL CHIPS. LOW LOGIC LEVELS ARE NORMALLY LESS THAN 0.2 VOLTS AND HIGH GREATER THAN 3.0 VOLTS. A LEVEL OF 0.4 VOLTS MAY INDICATE SHORTS BETWEEN OUTPUTS WHERE ONE IS TRYING TO PULL HIGH AND THE OTHER LOW. A LEVEL OF 1.2 VOLTS INDICATES AN OPEN CIRCUITED INPUT. NMOS CHIPS HAVE SIMILAR LOGIC LEVELS, WHILE PMOS CHIPS CAN PULL TTL INPUTS TO -0.6V WHERE THE INPUT CLAMP DIODE LIMITS THE VOLTAGE. DO NOT BE SURPRISED AT HOW STRANGE SOME OF THE WAVEFORMS ON THE BUS LOOK, SUCH AS THE DI LINES. THERE ARE PERIODS OF TIME DURING WHICH THE BUS IS NOT BEING ACTIVELY DRIVEN, AND THE VOLTAGE MAY DRIFT DUE TO RECEIVER INPUT CURRENT. ABNORMAL OPERATION IS INDICATED PRINCIPALLY BY ABNORMAL LOGIC LEVELS MAINTAINED CONSTANT FOR AT LEAST ONE CLOCK PERIOD (500 MICROSECONDS).

ONCE YOUR BASIC SYSTEM IS WORKING, CHECK OUT OF MEMORY BOARDS AND OTHER INTERFACES IS RELATIVELY STRAIGHTFORWARD USING THE MEMORY TEST PROGRAM IN THE MONITOR, OR SIMPLE DIAGNOSTIC ROUTINES YOU CAN PROGRAM IN MEMORY ON THE PROM/RAM BOARD. AFTER YOUR SYSTEM IS UP AND RUNNING, IT SHOULD BE QUITE RELIABLE. SINCE MOST MICROCOMPUTER SYSTEMS ARE MEMORY INTENSIVE, THE MEMORY IS THE MOST LIKELY SOURCE OF COMPONENT FAILURE. A SYSTEM WITH 32K OF STATIC MEMORY MAY CONTAIN 75% OF ITS COMPONENTS ON THE MEMORY BOARDS. IF A PROBLEM IS EXPERIENCED RUNNING A PROGRAM, FIRST SUSPECT THE MEMORY AND USE THE MONITOR TEST PROGRAM. WE HAVE YET TO EXPERIENCE A PROBLEM WITH OUR 8K MEMORY BOARDS THAT WAS NOT REVEALED BY THE TEST PROGRAM. IF YOU DO MUCH REARRANGING OF YOUR SYSTEM, IT IS A GOOD PRACTICE TO TEST MEMORY FOR A FEW SECONDS WHEN YOU FIRST TURN ON THE COMPUTER TO MAKE SURE THE BOARDS ARE ADDRESSED PROPERLY OR THAT THEY ARE IN THE COMPUTER. THIS MAY SAVE SOME HEAD SCRATCHING WHEN THE PROGRAM YOU HAVE JUST LOADED FAILS TO RESPOND TO YOUR EAGER KEYBOARD TOUCH. IF YOU SUSPECT TEMPERATURE SENSITIVE CHIPS, REMOVE THE COVER OF THE COMPUTER TO INTERRUPT AIR FLOW BETWEEN BOARDS. WE DO NOT RECOMMEND OBSTRUCTING THE AIR FLOW THROUGH THE COMPUTER BY PLACING A SHEET OF PAPER OVER THE LEFT SIDE. A FULL COMPUTER MAY DISSIPATE OVER 300 WATTS AND REACH UNACCEPTABLE TEMPERATURES IF NO AIRFLOW IS PERMITTED.

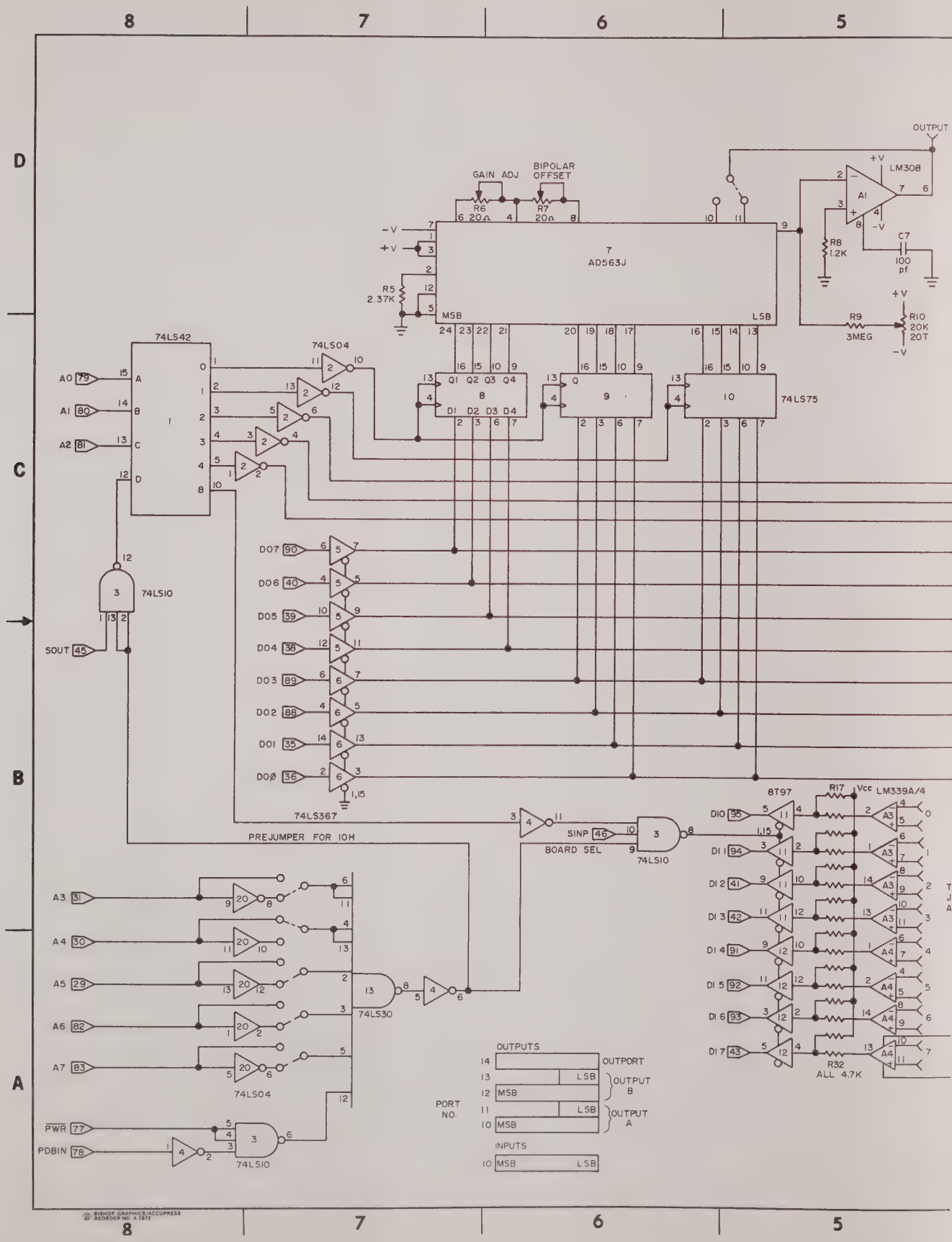
SIMPLE 12-BIT A/D CONVERSION ROUTINE

```
0010 STOR   DS    2      STORAGE FOR CONVERTED DIGITAL WORD
0020 CMP    EQU   10H    COMPARATOR INPUT PORT
0030 HDAC   EQU   12H    OUTPUT PORT 8MSB OF DAC
0040 LDAC   EQU   13H    OUTPUT PORT 4LSB OF DAC
0050        MVI   B,01   CHANNEL SELECT
0060        CALL  ADC
0070        SHLD  STOR   STORE CONVERTED VALUE
0080        RET          RETURN TO CALLING PROGRAM
0090 ADC    LXI   H,0    ZERO H,L
0100        MOV   D,H    ZERO D
0110        MOV   E,L    ZERO L
0120        STC          SET CARRY
0130 ADC1   MOV   A,D
0140        RAR          SHIFT SET BIT TO THE RIGHT
0150        MOV   D,A
0160        MOV   A,E
0170        RAR
0180        MOV   E,A
0190        ANI   08H    END OF CONVERSION
0200        RNZ
0210        MOV   A,E
0220        ORA   L
0230        MOV   L,A
0240        ANI   OFOH
0250        RLC
0260        RLC
0270        RLC
0280        RLC
0290        OUT   LDAC   OUTPUT 4 MSB TO DAC
0300        MOV   A,D
0310        ORA   H
0320        MOV   H,A
0330        OUT   HDAC   OUTPUT 8 LSB TO DAC
0340        IN    CMP    LOOK AT COMPARATOR OUTPUTS
0350        ANA   B      MASK CHANNEL
0360        JNZ   ADC1   COMPARATOR STATE?
0370        MOV   A,D    RESET TEST BIT
0380        XRA   H
0390        MOV   H,A
0400        MOV   A,E
0410        XRA   L
0420        MOV   L,A
0430        JMP   ADC1

$
```

3 CHANNEL A/D CONVERSION ROUTINE WITH 16 TIMES SUMMING

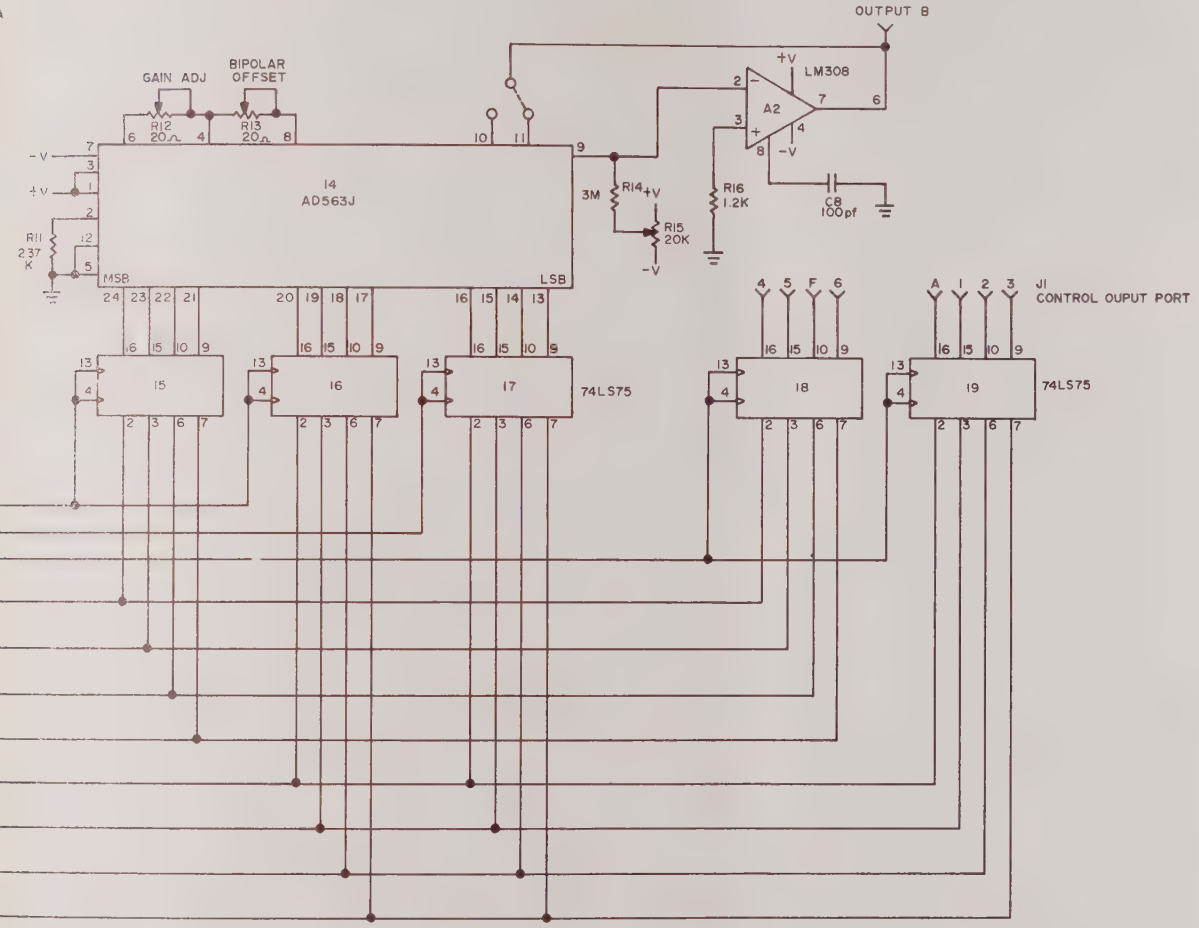
```
0010 STOR    DS     6        STORAGE FOR CONVERTED DIGITAL WORDS
0020 CNTL    DS     1
0030 COUNT   DS     1        COUNTER FOR ADC CALLS
0040         LXI    H,0      ZERO STORAGE AREAS
0050         SHLD   STOR
0060         SHLD   STOR+2
0070         SHLD   STOR+4
0080         MVI    B,01     CHANNEL SELECT
0090         MVI    A,0      COUNTER ZERO
0100         STA    COUNT
0110 LP      INR    A        COUNTER INCREMENT
0120         STA    COUNT
0130         CALL   ADC      ONE ADC CONVERSION
0140         CALL   SHIFT    FORMAT WORD FOR ADDITION
0150         LHLD   STOR     GET STORED SUM
0160         DAD    D        ADD RESULTS OF ADC
0170         SHLD   STOR     STORE NEW SUM
0180         LDA    COUNT    CHECK COUNTER
0190         CPI    10H      16 SUMS?
0200         JNZ    LP
0210         MVI    B,02     CHANNEL SELECT
0220         XRA    A
0230         STA    COUNT
0240 LP1     INR    A
0250         STA    COUNT
0260         CALL   ADC
0270         CALL   SHIFT
0280         LHLD   STOR+2
0290         DAD    D
0300         SHLD   STOR+2
0310         LDA    COUNT
0320         CPI    10H
0330         JNZ    LP1
0340         MVI    B,04     CHANNEL SELECT
0350         XRA    A
0360         STA    COUNT
0370 LP2     INR    A
0380         STA    COUNT
0390         CALL   ADC
0400         CALL   SHIFT
0410         LHLD   STOR+4
0420         DAD    D
0430         SHLD   STOR+4
0440         LDA    COUNT
0450         CPI    10H
0460         JNZ    LP2
0470         RET
0480 CMP     EQU    10H
0490 HDAC    EQU    12H
0500 LDAC    EQU    13H
0510 ADC     LXI    H,0      ADC CONVERSION ROUTINE
0520         MOV    D,H
0530         MOV    E,L
0540         STC
0550 ADC1    MOV    A,D
```

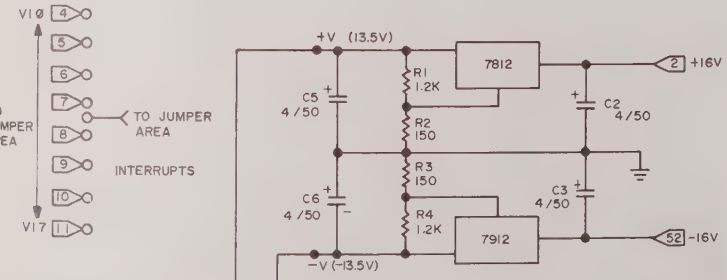
```
0560          RAR
0570          MOV    D,A
0580          MOV    A,E
0590          RAR
0600          MOV    E,A
0610          ANI    08H
0620          RNZ
0630          MOV    A,E
0640          ORA    L
0650          MOV    L,A
0660          ANI    0F0H
0670          RLC
0680          RLC
0690          RLC
0700          RLC
0710          OUT    LDAC
0720          MOV    A,D
0730          ORA    H
0740          MOV    H,A
0750          OUT    HDAC
0760          IN     CMP
0770          ANA    B
0780          JNZ    ADC1
0790          MOV    A,D
0800          XRA    H
0810          MOV    H,A
0820          MOV    A,E
0830          XRA    L
0840          MOV    L,A
0850          JMP    ADC1
0860 SHIFT    MOV    A,L        FORMATTING ROUTINE TO
0870          ANI    0F0H       PUT 12-BIT WORD IN
0880          RLC               PROPER ORDER FOR ADDITION
0890          RLC
0900          RLC
0910          RLC
0920          MOV    E,A
0930          MOV    A,H
0940          ANI    0FH
0950          RLC
0960          RLC
0970          RLC
0980          RLC
0990          ORA    E
1000          MOV    E,A
1010          MOV    A,H
1020          ANI    0F0H
1030          RLC
1040          RLC
1050          RLC
1060          RLC
1070          MOV    D,A
1080          RET

$0
```

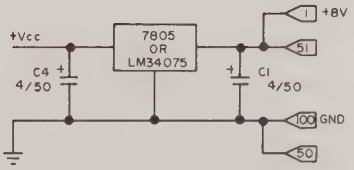


© BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-7872

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED







UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: DECIMALS .XX ± .02 .XXX ± .010 ANGLES ±½°	CONTRACT NO.	VECTOR GRAPHIC INC. 790 HAMPSHIRE RD. WESTLAKE VILLAGE CALIFORNIA, 91361	
MATERIAL	APPROVALS / DRAWN GEORGE STROUD / CHECKED	DATE 2-12-78 / 2-22-78	PRECISION ANALOG I/O BOARD
FINISH			SIZE D / CODE IDENT NO. / DRAWING NO.
NEXT ASSY / USED ON			SCALE / SHEET OF
APPLICATION	DO NOT SCALE DRAWING		

Z80 BOARD, REVISION 2

USERS MANUAL

Revision A

May 17, 1979

*Z-80 is a registered trademark of Zilog Corporation.

Copyright 1979 Vector Graphic Inc.

Copyright 1979 by Vector Graphic Inc.
All rights reserved.

Disclaimer

Vector Graphic makes no representations or warranties with respect to the contents of this manual itself, whether or not the product it describes is covered by a warranty or repair agreement. Further, Vector Graphic reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Vector Graphic to notify any person of such revision or changes, except when an agreement to the contrary exists.

Revisions

The date of release and revision letter of each page herein appears at the bottom of each page. The revision letter such as A or B changes if the manual has been improved but the product itself has not been significantly modified. The date of release and revision letter on the Title Page corresponds to that of the page most recently revised. When the product itself is modified significantly, the product will get a new revision number, as shown on the manual's title page, and the manual will revert to revision A, as if it were treating a brand new product. EACH MANUAL SHOULD ONLY BE USED WITH THE PRODUCT IDENTIFIED ON THE TITLE PAGE.

Rev. 2-A 5/17/79

TABLE OF CONTENTS

Section — Page

Repair Agreement
Table of Contents

I. Introduction

1.1 Specifications....1-1
1.2 Description of the Z-80 Revision 2....1-2

II. Users Guide

2.1 System Compatibility....2-1
2.2 Using Dynamic Memory - Bus Lines 55 and 66....2-1
2.3 Not Using Dynamic Memory....2-2
2.4 High Order Address Lines During I/O....2-2
2.5 Running the System at 2 or 4 MHz....2-3
2.6 Automatic Wait States for Slow Memories....2-3
2.7 Generating MWRITE....2-3

III. Theory of Operation

3.1 Power....3-1
3.2 Clock Generation....3-1
3.3 High Order Address Lines During I/O....3-1
3.4 Control Signals....3-2
3.5 Wait States....3-2
3.6 Miscellaneous Logic....3-3

Figures....3-4

IV. Schematics

Board Layout....4-1
Schematic Errata....4-2
Schematic....4-3

I. INTRODUCTION

1.1 SPECIFICATIONS

Bus Compatibility	S-100
Processor	Mostek Z-80A (MK-3880N)
Number of Data Bits	8
Number of Address Bits	16
Instructions	158, including all 78 8080 instructions
Clock speed	2 or 4 MHz, jumper selectable Standard: 4 MHz
Interrupts	All three Z-80 modes S-100 vectored interrupt lines are not handled
I/O devices	256 I/O addresses
Dynamic RAM	Supports dynamic memory by sending Z-80 $\overline{\text{RFSH}}$ on bus line 66; and using fast reset received on line 55
Static RAM	Jumper option to generate reset on Z-80 board instead of fast reset on line 55
MWRITE	Jumper option to generate MWRITE on board Standard: option enabled
Wait state generation for memories slower than about 300 ns.	Jumper option to generate one wait state each time memory is addressed Standard: option not enabled
High address lines during I/O	For peripherals that need it, high address lines mirror low address lines during I/O
Bus load	1 low power TTL load on all inputs
Buffering	All outputs buffered
Card extractors	Standard
Power	+8Vdc @ 450 mA (Typ)

1.2 DESCRIPTION OF THE Z-80 REVISION 2 BOARD

The Vector Graphic Z-80 board is designed around the powerful Z-80 microprocessor chip. The board is fully S-100 bus compatible, providing interchangeability between systems. All input and output lines are fully buffered. Loading on the bus is no more than 1 TTL load.

In addition, the board has the necessary circuitry to work well with dynamic memories, such as Vector Graphic's 48K board.

Simplicity of design has been stressed to enhance reliability of operation by the use of MSI and LSI integrated circuits. This goal has been achieved while not sacrificing any performance factors.

II. USERS GUIDE

For the location of jumper areas, refer to the silk-screen diagram on page 4.1

2.1 SYSTEM COMPATIBILITY

The Z-80 Board is designed for operation in an S-100 system. It generates ALL the S-100 signals originally generated by 8080 CPU boards following the S-100 definition, and accepts as input the S-100 signals intended as input to an 8080 processor board, with two exceptions: 1) Vectored interrupt request lines (S-100 lines 4 - 11) are not received or handled in any way by the board. 2) The timing of some signals, particularly PSYNC and the status signals, is slightly different than that produced by 8080 boards, but the differences are usually inconsequential. Also, a few additional lines are used, required for dynamic memory in a Z-80 based system, and are explained below.

For power, the board only requires the +8V unregulated source on bus line 1.

2.2 USING DYNAMIC MEMORY - BUS LINES 55 AND 66

The Z-80 Board as manufactured is configured for use with dynamic memory boards. The following paragraphs will explain how.

The Z-80 chip provides a $\overline{\text{RFSH}}$ signal on pin 28 used to refresh dynamic memory, thus enabling easier use of dynamic memory in a system. This signal is buffered directly onto line 66 of the bus. Thus appropriately designed dynamic memory boards can be used in conjunction with this Z-80 board, as in the Vector Graphic MZ computer. For use of the $\overline{\text{RFSH}}$ signal in designing dynamic memory components, refer to Z-80 data books.

Dynamic memory requires that each byte be read and rewritten continuously. If the processor is halted for too long an interval, memory data can be lost. The $\overline{\text{RESET}}$ signal which is created by circuitry on the Z-80 board from an incoming $\overline{\text{PRESET}}$ signal on line 75 is too long. Hence, for use with dynamic memory, THE BOARD IS SHIPPED WITH THIS CIRCUITRY DISABLED. Instead,

the dynamic memory board is expected to create a fast reset signal on bus line 55, which is then used as input to the processor $\overline{\text{RESET}}$ pin.

If you have obtained a particular Z-80 board which has had the $\overline{\text{RESET}}$ circuit re-enabled, as described in the following section, then simply reverse the steps in the following section in order to use the board with a dynamic memory board that generates a fast reset on bus line 55.

2.3 NOT USING DYNAMIC MEMORY

If the system is not using the Vector Graphic 48K Dynamic Memory Board or other dynamic memory board which generates a fast reset signal, then you must enable the generation of $\overline{\text{RESET}}$ by the Z-80 board. Once enabled, the board will generate $\overline{\text{RESET}}$ using the $\overline{\text{PRESET}}$ signal on bus line 75, or, when the system is first turned on, using a resistor-capacitor circuit on the board. The necessary circuitry already exists on the board, except as follows. Take the following steps:

a. Install a 100 ohm 1/4 watt resistor in the spot marked to the left of U16.

b. Make sure there is capacitor installed in a marked location very close to the lower left hand corner of the Z-80 chip. If not, install a 22 or 25 MFD 12 to 16 V axial electrolytic capacitor at that spot, with the plus end to the LEFT.

c. Install a jumper across area D, found at the bottom of the board, just above the middle of the end connector. This jumper connects bus line 75 - $\overline{\text{PRESET}}$, to the reset circuitry.

d. Cut the jumper connecting pad E and pad F. It is usually installed on the back of the board. Pad E is at the bottom of the board, on the left side, and F is between U6 and U7. This jumper connects bus line 55 to the Z-80 $\overline{\text{RESET}}$ pin.

2.4 HIGH ORDER ADDRESS LINES DURING I/O

The Z-80 microprocessor presents the peripheral address byte on the low order byte of address (bits 0-7) during an I/O operation. However, some peripheral boards require the address byte to be present on the high order

byte. In order to maintain compatibility in all cases, logic has been included to "mirror" the peripheral address on the low order byte also onto the high order byte during an I/O operation.

2.5 RUNNING THE SYSTEM AT 2 OR 4 MHz

Jumper area: A

Connections as manufactured: none

Function: selects 2 or 4 MHz CPU operation. The Z-80 chip is capable of 4 MHz operation, but some peripheral boards are not. The board is shipped for operation at 4 MHz.

Options: to operate at 2 MHz, install a jumper in area A.

2.6 AUTOMATIC WAIT STATES FOR SLOW MEMORIES

Jumper area: B

Connections as manufactured: none

Function: When this jumper is connected, the Z-80 board will cause one wait state to be inserted in each memory access instruction. This is needed when using memories in the system slower than about 300 ns.

Options: to enable the automatic wait state, add a the jumper in area B.

2.7 GENERATING MWRITE

Jumper area: C

Connections as manufactured: jumper installed.

Function: when connected, the Z-80 board will generate the MWRITE signal (S-100 line 68). You will want to disconnect this jumper if there is some other source of MWRITE in the system, such as a front panel.

Options: if MWRITE generation is not wanted, remove the jumper in area C.

III. THEORY OF OPERATION

3.1 POWER

The Vector Graphic Z-80 board requires only a single 5 volt DC power source and regulates the raw +8V with a 7805 regulator. Ample bypass and filtering is provided by electrolytic and ceramic capacitors distributed on the board.

3.2 CLOCK GENERATION

See Figure 1. This board has been designed to operate at both 2 MHz and 4 MHz. Although the Z-80 requires only a single phase clock for operation, the Vector Graphic Z-80 board has circuitry to generate the Ø1, Ø2 and $\overline{CLK}$ signals required by the S-100 bus and components that interface to it.

The basic source of all the clock signals is an on-board 8 MHz crystal oscillator. The 8 MHz signal is applied to divide by 2 and divide by 4 logic formed by flip-flops U8a and U8b. The 2 MHz output at U8-8 is used for the $\overline{CLK}$ signal.

The 8 MHz clock and 4 MHz output at U8-6 are applied to gating logic to select, per jumper position, 2 or 4 MHz system operation. The subsequent output of this gating logic is applied to U7-11. By using both the Q and $\overline{Q}$ outputs of U7 the two phase clock relationship is achieved and by gating the input clock with U7-9 the required delay between clocks is achieved. Ø1 and Ø2 signals are applied to bus lines 24 and 25. The Z-80 is clocked by Ø2.

3.3 HIGH ORDER ADDRESS LINES DURING I/O

As noted in the User's Guide, to satisfy the needs of some peripheral boards, logic has been included to "mirror" the peripheral address on the low order byte also onto the high order byte during an I/O operation. This is implemented by "wire or"ing the buffered low order address bits with the buffered high order address bits. The tri-state buffers used are enabled during an I/O operation only.

3.4 CONTROL SIGNALS

The Z-80 microprocessor has five basic control lines which are used to read or write data to and from memory or I/O devices. these signals are:

1. $\overline{\text{MREQ}}$ - output, active low. The memory request signal indicates that the address bus holds a valid address for a memory reference operation.

2. $\overline{\text{M1}}$ - output, active low. $\overline{\text{M1}}$ indicates that the current machine cycle is the op code fetch cycle of an instruction execution.

3. $\overline{\text{IORQ}}$ - output, active low. The $\overline{\text{IORQ}}$ signal indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. An $\overline{\text{IORQ}}$ signal is also generated when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus.

4. $\overline{\text{RD}}$ - output, active low. $\overline{\text{RD}}$ indicates that the Z-80 wants to read data from memory or an I/O device.

5. $\overline{\text{WR}}$ - output, active low. $\overline{\text{WR}}$ indicates that the Z-80 data bus holds valid data to be stored in the addressed memory or I/O device.

By logical manipulation of the above signals the S-100 bus type signals are created. Thus, SMEMR is the logical "AND" of $\overline{\text{RD}}$ and $\overline{\text{MREQ}}$, MWRITE is $\overline{\text{WR}}$ and $\overline{\text{MREQ}}$ (produced by NORing $\overline{\text{WR}}$ and SOUT), SOUT is $\overline{\text{WR}}$ and IORQ, SINP is $\overline{\text{RD}}$ and IORQ, PDBIN is $\overline{\text{RD}}$, and PWR is $\overline{\text{WR}}$. All the previous listed signals are fully buffered to the bus.

The Z-80 microprocessor does not have an equivalent output to PSYNC so circuitry has been included to emulate this function. $\overline{\text{MREQ}}$ and $\overline{\text{IORQ}}$ are logically "OR"ed and applied to the D input of a flip-flop clocked by the system clock. Using the Q output, the PSYNC signal is true for 500 nsec at 2 MHz and 250 nsec at 4 MHz. Refer to figure 2.

3.5 WAIT STATES

See figure 3. The Z-80 CPU board provides for a jumper selectable automatic wait state. For many applications, it may be desirable to use slow memories. The wait logic on the board allows the Z-80 to operate with slower memory. The user may select this wait state generation if required.

The automatic wait state is created by setting U12-9 when either MREQ or IORQ are true (if the appropriate jumper is installed). The following clock time U12 is set to the Q state, thus turning off the wait signal. This circuit guarantees that wait will be true when the Z-80 looks for a wait.

As with 8080-based CPU's, external devices can cause the computer to enter a wait state by applying the appropriate signal to the PRDY bus line.

3.6 MISCELLANEOUS LOGIC

Logic is provided on the board to cause the PINTE bus line to set and reset upon the execution of EI (enable interrupt) and DI (disable interrupt) instructions, respectively.

The Z-80 board includes the capability to use a fast reset signal on bus line 55 to produce the RESET input to the Z-80. Alternately, as described in the User's Guide, existing circuitry on the board can be used to generate RESET, using the PRESET input on line 75 of the bus. If this option is used, RESET is also generated on board when the system is initially turned on. (Using an RC time delay connected to VCC, the Z-80 RESET input is low for a short time when the system is first powered up.) Regardless of how it is generated, the RESET signal is used to generate the power on clear (POC) output on bus line 99.

FIGURE 1

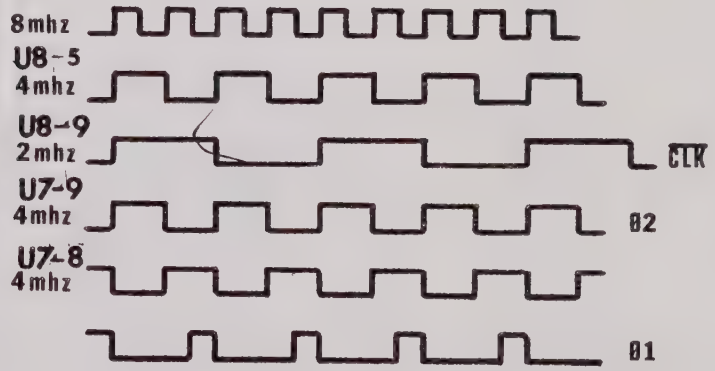


FIGURE 2

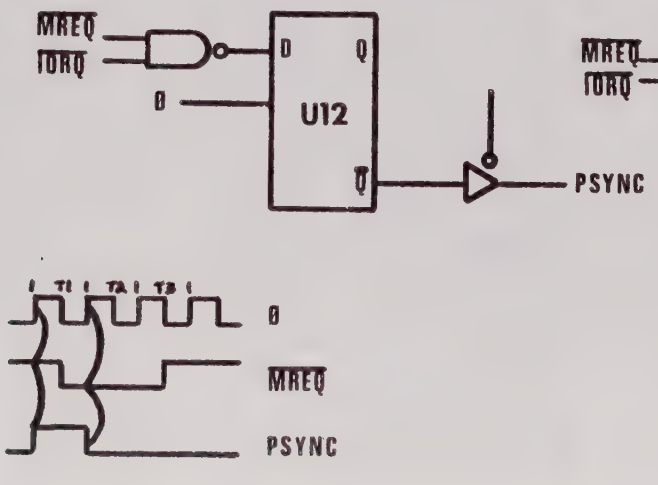
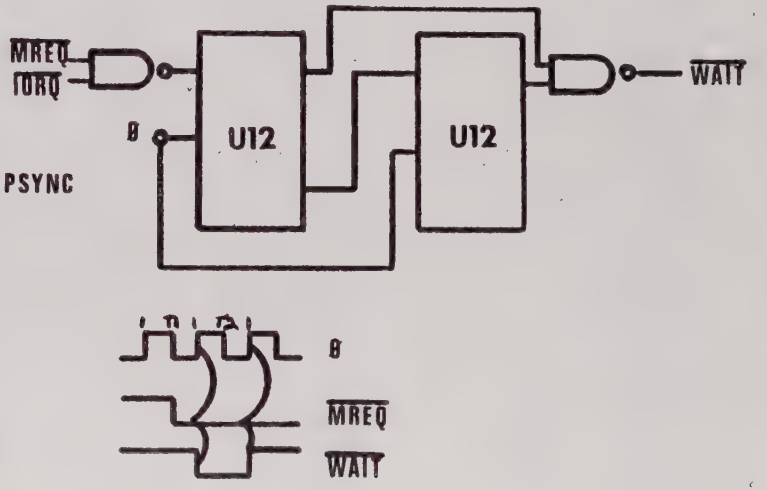
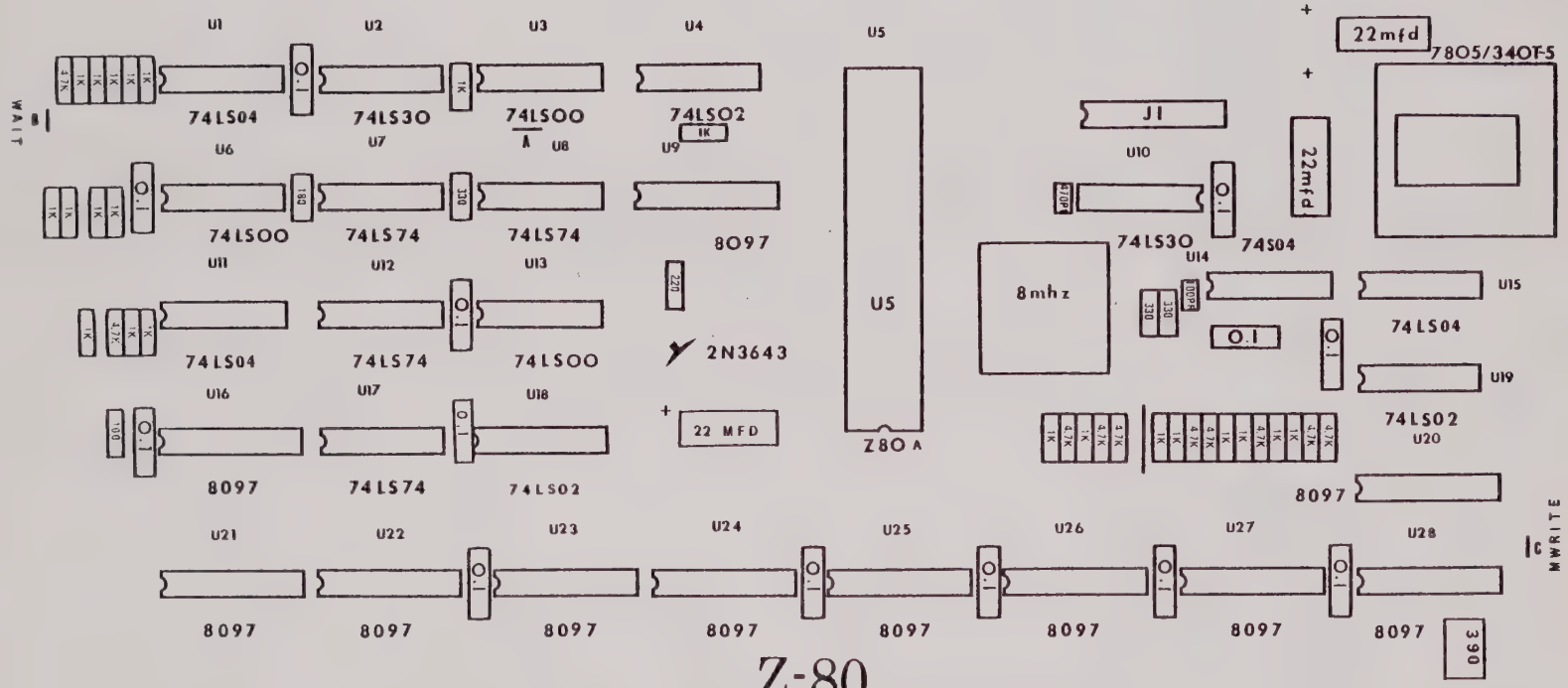


FIGURE 3



10.000

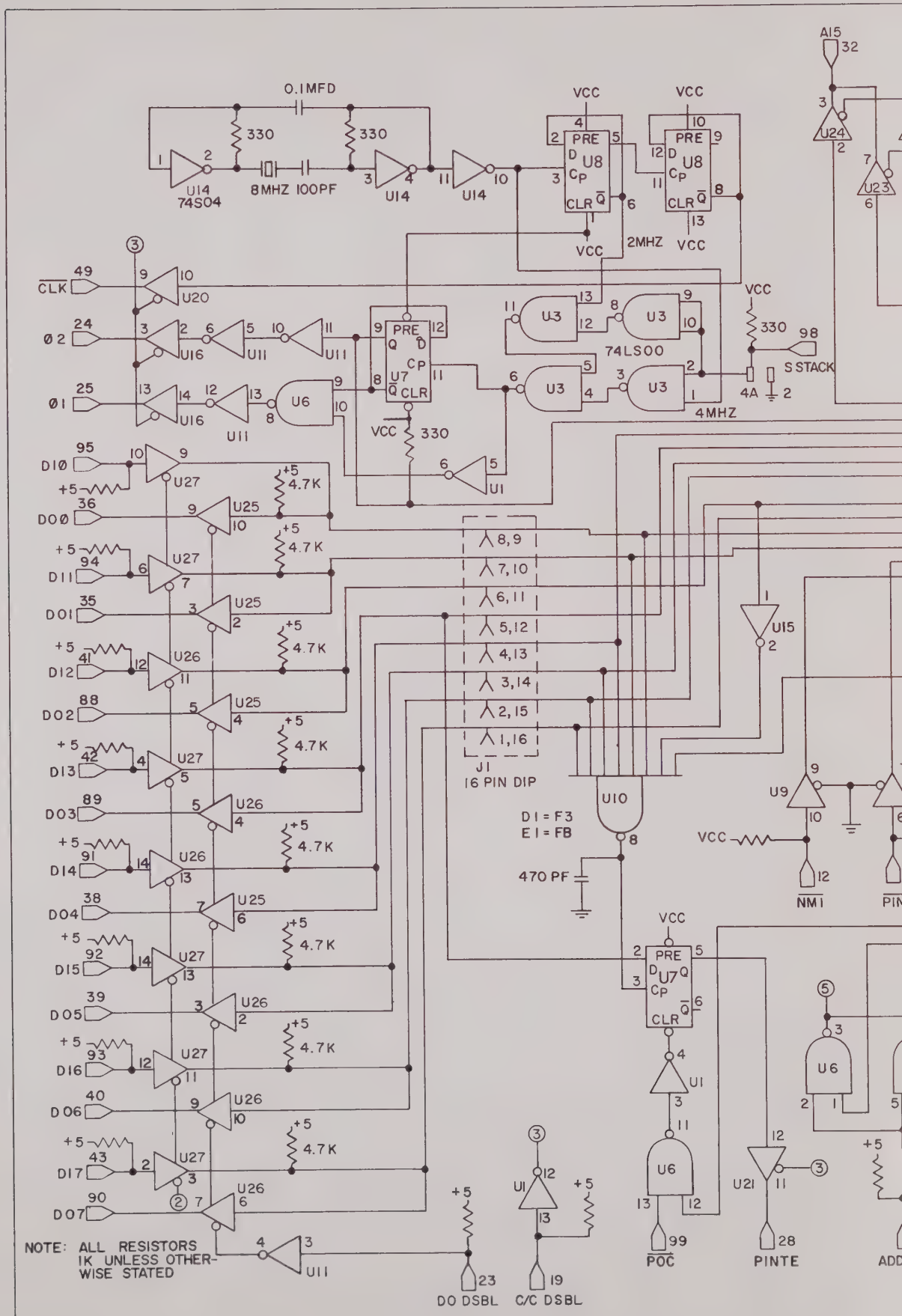

U1
U2
U3
U4
U5
74LS04
74LS30
74LS00
74LS02
U6
U7
U8
U9
U10
74LS00
74LS74
74LS74
8097
J1
22mfd
7805/340T-5
74LS30
74S04
U11
U12
U13
U14
U15
74LS04
74LS74
74LS00
2N3643
8mhz
74LS04
U16
U17
U18
U19
22 MFD
Z80A
74LS02
U20
8097
74LS74
74LS02
U21
U22
U23
U24
U25
U26
U27
U28
8097
8097
8097
8097
8097
8097
8097
8097
WAIT
MWRITE
Z-80
VECTOR GRAPHIC INC.

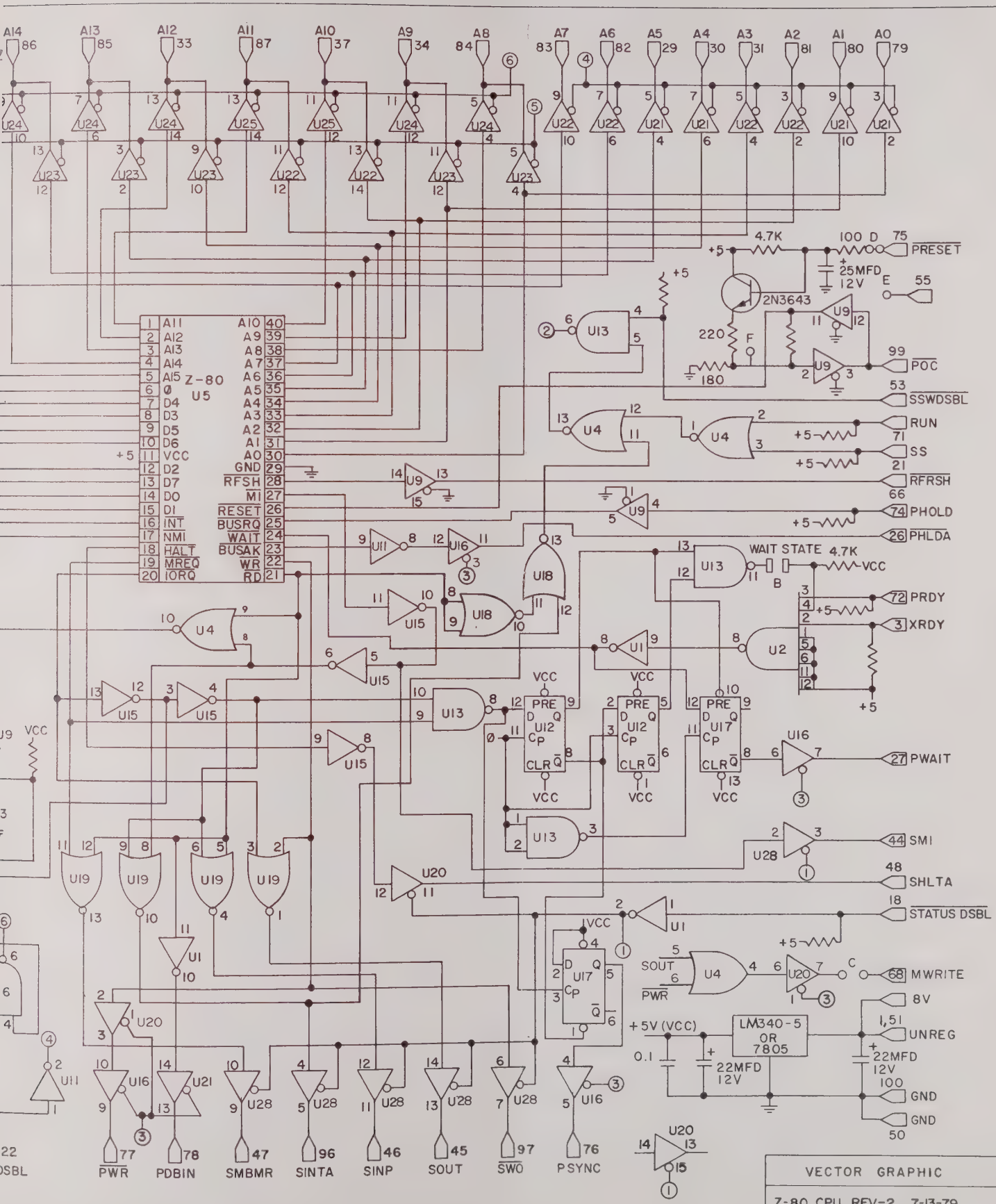
SCHEMATIC ERRATA

1. There should be no connection across jumper area D, in the upper right-hand corner of the schematic. The 100 ohm resistor and 25 MFD capacitor connected to it are not shipped on the board, but their places on the board has been set aside. The board is shipped with a jumper from pad E to pad F, in the upper right-hand corner of the schematic.

2. In the lower right-hand quadrant of the schematic, there is an inverter labeled U19, having pins 11 and 12. This should be labeled U20.

3. Jumper are C, near the lower right-hand corner of the schematic, should be shown connected.


0.1MFD
330
330
8MHZ 100PF
U14
74S04
U14
U14
VCC
VCC
U8
PRE
CLR
2MHZ
VCC
A15
32
U24
U23
CLK
49
U20
Ø2
24
U16
U11
U11
Ø1
25
U16
U11
U6
U7
PRE
CLR
VCC
330
U1
U3
U3
U3
U3
74LS00
VCC
330
98
S STACK
4A
2
4MHZ
D10
95
U27
+5
4.7K
D00
36
U25
D11
94
U27
D01
35
U25
D12
41
U26
D02
88
U25
D13
42
U27
D03
89
U26
D14
91
U26
D04
38
U25
D15
92
U27
D05
39
U26
D16
93
U27
D06
40
U26
D17
43
U27
D07
90
U26
U11
NOTE: ALL RESISTORS 1K UNLESS OTHERWISE STATED
8,9
7,10
6,11
5,12
4,13
3,14
2,15
1,16
J1
16 PIN DIP
U10
D1 = F3
E1 = FB
470 PF
U15
U9
VCC
NMI
PIN
VCC
U7
PRE
CLR
U1
U6
U6
99
POC
U21
28
PINTE
ADD
U1
+5
23
DO DSBL
19
C/C DSBL


A14 86
A13 85
A12 33
A11 87
A10 37
A9 34
A8 84
A7 83
A6 82
A5 29
A4 30
A3 31
A2 81
A1 80
A0 79
Z-80 U5
A11
A12
A13
A14
A15
Ø
D4
D3
D5
D6
VCC
D2
D7
DO
DI
INT
NMI
HALT
MREQ
IORQ
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
GND
RFSH
MI
RESET
BUSRQ
WAIT
BUSAK
WR
RD
4.7K
100 D
25MFD 12V
2N3643
220
180
PRESET 75
E 55
F
99 POC
53 SSWDSBL
RUN 71
SS 21
RFRSH 66
PHOLD 74
PHLDA 26
WAIT STATE 4.7K
VCC
B
PRDY 72
XRDY 3
PWAIT 27
SMI 44
SHLTA 48
STATUS DSBL 18
MWRITE 68
8V
UNREG 1,51
GND 100
GND 50
+5 (VCC)
0.1
LM340-5 OR 7805
22MFD 12V
SOUT
PWR
PRE
CLR
PWR 77
PDBIN 78
SMBMR 47
SINTA 96
SINP 46
SOUT 45
SWO 97
PSYNC 76
VECTOR GRAPHIC
Z-80 CPU REV-2 7-13-79

PROM/RAM III BOARD

Revision 1

and

PROM PROGRAMMING PROGRAM

Revision 1

USERS MANUAL

Revision A

July 16, 1979

Copyright 1979 Vector Graphic Inc.

Copyright 1979 by Vector Graphic Inc.
All rights reserved.

Disclaimer

Vector Graphic makes no representations or warranties with respect to the contents of this manual itself, even if the product it describes is covered by a warranty or repair agreement. Further, Vector Graphic reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Vector Graphic to notify any person of such revision or changes, except when an agreement to the contrary exists.

Revision Numbers

The date and revision of each page herein appears at the bottom of each page. The revision letter such as A or B changes if the manual has been improved but the product itself has not been significantly modified. The date and revision on the Title Page corresponds to that of the page most recently revised. When the product itself is modified significantly, the product will get a new revision number, as shown on the manual's title page, and the manual will revert to revision A, as if it were treating a brand new product. THIS MANUAL SHOULD ONLY BE USED WITH THE PRODUCT(S) IDENTIFIED ON THE TITLE PAGE.

Rev. 1-1-A 7/16/79

TABLE OF CONTENTS

Section — Page

Repair Agreement
Table of Contents

I. Introduction

1.1 Specifications....1-1
1.2 Description of the PROM/RAM III Board....1-3

II. Users Guide

2.1 PROM Selection and Use....2-1
2.2 RAM....2-1
2.3 Block A and Block B - General....2-2
2.4 Block A....2-3
2.5 Block B....2-3
Figures 2 & 3 - Examples of Block B Configurations....2-5
2.6 Block Select Addressing....2-6
2.7 PROM/Scratchpad Memory Invert....2-6
2.8 RAM Memory Address Select in Block B....2-7
2.9 Disable 3K of Address Space in Block B....2-7
2.10 Power-on/Reset Jump - Description....2-8
2.11 Use PRESET or POC for Power-on/Reset Jump....2-8
2.12 Phantom Generated if Power-on/Reset....2-9
2.13 Jump to PROM/RAM III Board if Power-on/Reset....2-9
2.14 Block Swap....2-10
2.15 Disable Power-on/Reset Response....2-10
2.16 MWRITE....2-10
2.17 Wait State Generation....2-11

Tables 1, 2, and 3....2-12

2.18 Programming a PROM - Normal Procedure....2-13
2.19 Writing a PROM Programming Program....2-16
2.20 Re-assembling the PROM Programming Program....2-17
2.21 PROM Programming Program Listing....2-18 - 2-24

III. Theory of Operation

3.1 Addressing....3-1
3.2 Data Input/Output....3-2
3.3 Control Signals....3-2
3.4 PROM Programming....3-3
3.4 Power Supplies....3-4

IV. Schematics

Board Layout.....4-1
Schematic Errata.....4-2
Schematic.....4-3

I. INTRODUCTION

1.1 SPECIFICATIONS

Bus Compatibility	S-100
Memory Capacity	RAM: 1K, included with the board PROM: Sockets for 12 PROMs.
PROM Programming	Can program 2708 or 2704 EPROMs
PROM Programming Program	Listing included in manual Executable version on MDOS System Diskettes 8.4 and later.
PROMs Included with Board	NONE
Memory Speed	RAM: 300 ns. PROM: User selected (450 ns. typ)
Memory Types	RAM: 2114 static PROM: 2708 (1K each) or 2704 (1/2K each)
Board Addressing	Two blocks (A and B) are separately addressed Block A has 8 PROM sockets Block B has 4 PROM sockets and 1K RAM
Addressing Options (jumper)	Base address of the two 8K blocks Block B PROM at top or bottom of block Address of 1K RAM within remaining 4K Disable unused 3K, for use by other boards
Standard Addressing	Block A: disabled Block B base address: C000H Block B PROMs: C000H - CFFFH Block B RAM: DC00H - DFFFH Block B disabled 3K: D000H - DBFFH
Standard Location of Systems Monitor PROM	C000H

(continued on back)

Power-on/Reset Jump	PRESET or POC causes jump to board
Power-on/Reset Jump Options (jumper)	Use PRESET or POC Jump to first instruction of Block A or B. Disable phantom generation Disable jump to on-board memory
Standard Power-on/Reset Jumpers	POC is used Jump to beginning of Block B Phantom and jump to on-board both enabled
MWRITE	Jumper option to generate MWRITE on board Standard: option not enabled
Wait state generation	Jumper option to generate one wait state each time board is addressed Standard: option not enabled
Bus load	1 TTL load on all inputs
Card extractors	Standard
Power	+8Vdc @ 450 mA (Typ) +18Vdc @ (depends on quantity of PROM) -18Vdc @ (depends on quantity of PROM)

1.2 DESCRIPTION OF THE PROM RAM III BOARD

Vector Graphic's PROM RAM III Board is a versatile, S-100 bus compatible, high density memory board combining the memory technologies of erasable programmable read only memories (EPROMs) and high speed random access memory (RAM). Of unique value, one of the PROM sockets on the board can be used to program a 2708 or 2704 EPROM, enabling any owner to create PROM-based software for use on this board or in any other microprocessor device. 1K of RAM is provided on the board, but no PROMs are included with purchase. The software which is used to program PROMs is provided as a listing in this manual, and is included on disk with all Vector Graphic systems shipped with this board.

By combining the use of MSI decoding logic and unique addressing features, a wide range of applications requirements may be met by this memory board. The addressing flexibility is as follows. The board offers two independently addressable 8K blocks of memory (A and B). You use jumpers to specify the two separate 8K addressing spaces assigned to these blocks. Block A can be used for up to 8K of PROM. Block B contains 1K of on-board RAM plus up to 4K of PROM.

For block B, you use jumpers to specify whether the PROM is at the top or the bottom of the 8K allocation, and then, within the remaining 4K, where the 1K of RAM is addressed. Once this is done, there are also jumper options for DISABLING some or all of the remaining 3K of addressing space allocated to block B, so that other boards in the system can use those addresses.

The addressing spaces are fully utilized if 2708 1K PROMs are used. If 2704 1/2K PROMs are used, then every other 1/2K of PROM allocation will be used, with 1/2K gaps between. Other features offered by the board are: jump on power-on or reset to on-board memory, with phantom generated to temporarily disable other memory boards, and a jumper option to use PRESET instead of POC to cause this jump; jumper option for on-board generation of the S-100 MWRITE signal; and a jumper option to generate a one-cycle wait-state each time the board is addressed.

Full buffering of all inputs and outputs is provided to minimize loading of the system S-100 bus to at most one TTL load. On-board power regulation and filtering is provided using IC regulators and heat sinks for power dissipation. Careful attention to good design practice and an awareness of the need for flexibility has resulted in a reliable board useful in a wide variety of systems and applications.

II. USERS GUIDE

This Users Guide begins with a description of the amount and kind of PROM which can be used on this board, followed by a description of the RAM included with the board, then a detailed description of the various options you have for addressing the PROMs and the RAM. Read it before attempting to re-jumper the board addressing. Following this section are a description of each of the jumper options possible on the board, including addressing options, power-on/reset jump, MWRITE input, and wait state generation. The diagrams of jumper pads show each of the pads as it is pre-jumpered at the factory. The guide ends with instructions for operating the PROM programming software provided with the board, as well as instructions for writing your own if desired. The listing of the program is provided.

2.1 PROM SELECTION AND USE

A maximum of 12K bytes (where K = 1024) of 2708 type PROMs may be installed in available sockets on the board. NO PROMS ARE INCLUDED WITH PURCHASE OF THE BOARD ALONE. Jumpers are used to determine where the PROMs are addressed.

The following discussion assumes that 2708 type PROMs (having 1K of 8-bit bytes each) are used. If 2704 PROMs (having 1/2K bytes each) are used, the issues are the same; the only difference is that wherever a 2704 PROM is used, there will be 1/2K bytes of PROM accessible by the system, followed immediately by a 1/2K gap which will not contain any memory at all.

The numbers 2708 and 2704 are Intel generic part numbers. Many other manufacturers make equivalents, with 2708 or 2704 as part of their proprietary part number. All 2708 or 2704 pin for pin equivalents can be used on this board.

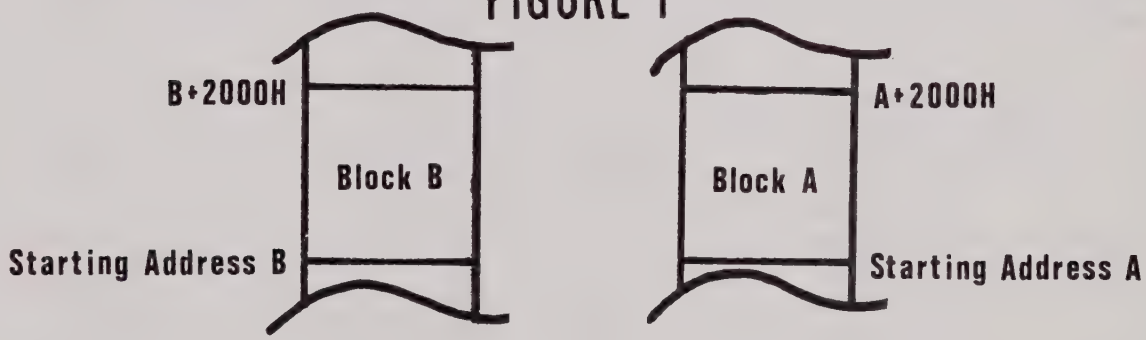
2.2 RAM

In addition to the PROM sockets, there is 1K of static RAM on the board, which IS included with purchase of the board alone. Jumpers are used to determine where this 1K of RAM is addressed.

2.3 BLOCK A AND BLOCK B - GENERAL

To begin specifying the addresses for the memory, there are two separately addressable blocks of memory space available on the board, called blocks A and B. Jumpers are used to specify what the base address is for each of these two blocks, within a 64K total memory space. Alternately, one (or both) blocks can be disabled completely. Jumper area F is normally used to specify the base address of (or disable) block A and jumper area E is normally used to specify the base address of (or disable) block B. If a block is not disabled, then that block will occupy exactly 8K bytes of memory, beginning at its base address. This is true for both blocks, as shown in Figure 1.

FIGURE 1



Note that both blocks together occupy 16K of memory. However, there are only 12 sockets for PROMs, and only 1K of RAM on the board, totalling 13K. What happens if the processor addresses memory in the remaining 3K portion? This memory space is NOT necessarily empty. A set of jumpers is provided which in effect specify that the unused 3K, within the 16K, is not on the PROM RAM III board at all, and therefore may be used on other boards.

It must be emphasized that except for the 3K specified as unused by jumper, the addresses assigned to the board for blocks A and B cannot be used by any other board, even if some of the PROM sockets are left empty. However, remember that you may choose not to use one (or both) of the blocks at all, by disabling it completely in jumper areas E and F. If you do this, then the corresponding memory space CAN be assigned to another board, and no space is wasted.

If the jumpers in area G are switched from the way the board is normally shipped, then the base address of block A will be controlled by jumper area E and the base address of block B will controlled by jumper area F, instead of the other way around. If this is done, then the address which is accessed for power-on jump will also be switched, becoming the first address in block A instead of the first address in block B. This is the purpose for using this option. (See Section 2.14) For simplicity of language, the Users Guide is written assuming that jumper area G is left as manufactured.

2.4 BLOCK A

Block A refers to the 8 PROM sockets at the top of the board (labeled 0 through 7). Insert PROMs which you want in block A into these sockets. Socket 0 corresponds to the 1K block beginning at the base address of block A. Socket 1 corresponds to the next 1K and so on, as shown in the following table:

Hexadecimal Address Relative to Base Address ("A") of Block A	Socket
A + 1C00H	7
A + 1800H	6
A + 1400H	5
A + 1000H	4
A + C00H	3
A + 800H	2
A + 400H	1
A	0

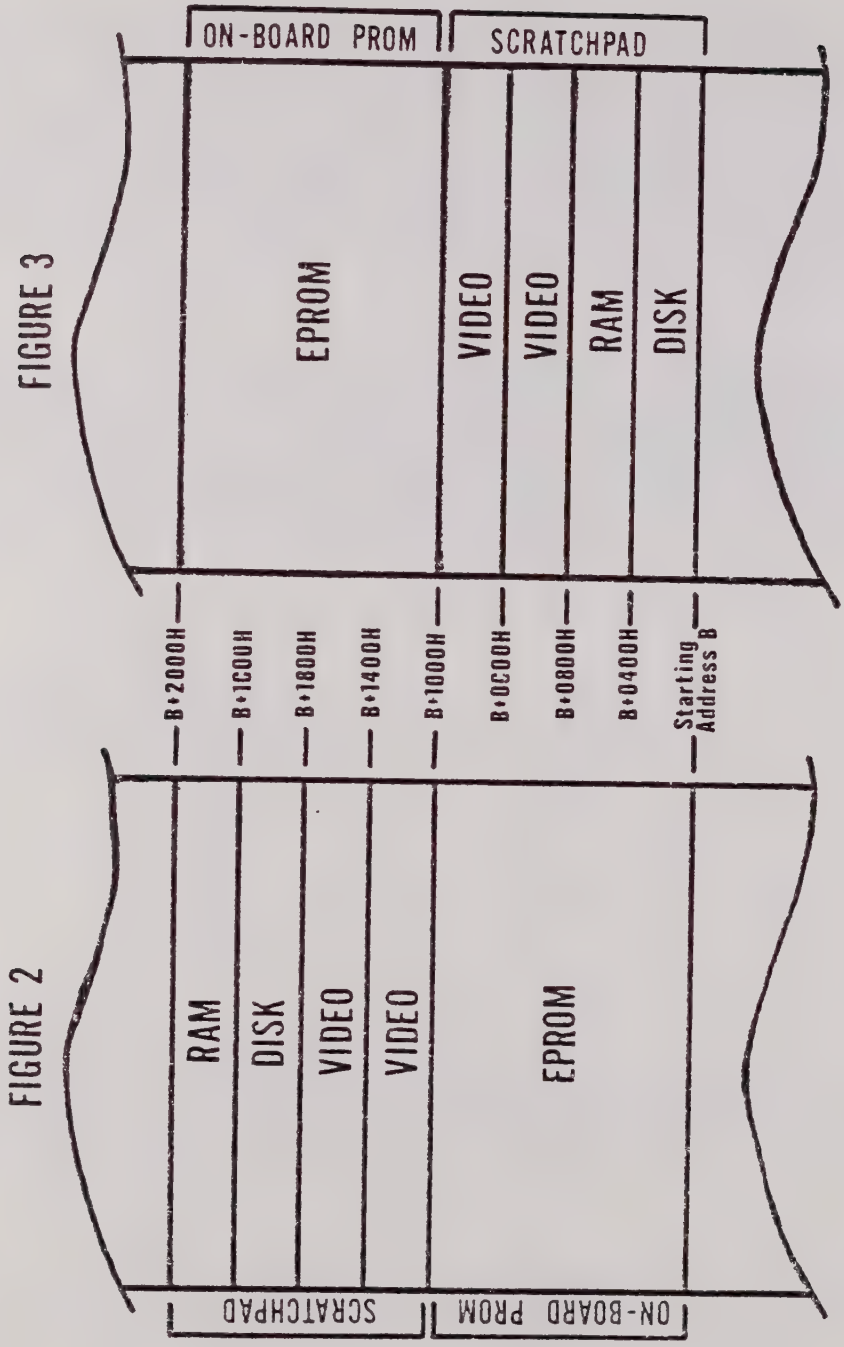
Jumper area F is normally used to determine the base address of block A, or to disable block A. When the board is sold, jumper area F is pre-wired to disable block A. No particular base address is thus specified until you install the jumpers.

2.5 BLOCK B

Block B includes the lower four PROM sockets on the board, labeled 8 through 11. The other 4K in block B is filled with the 1K of RAM on the board, plus the 3K of address space which can be, at you discretion, returned for use by

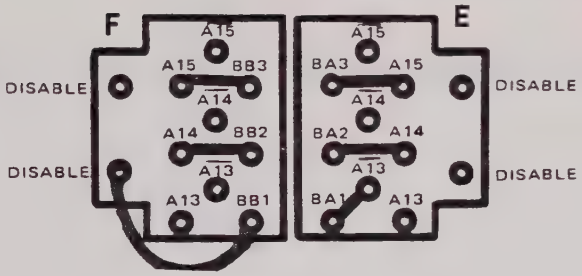
other boards. The way you specify the address spaces within block B is as follows: First, you specify the base address of Block B using jumper area E (or you specify in area E that the block is disabled). If it is not disabled, then you use jumper area J to specify whether the 4K of PROM occupies the top or the bottom 4K of the block. These are the only two choices. The board is pre-jumpered so that the PROM occupies the lower 4K. Then, you specify using jumper area I which 1K within the other 4K is used for the on-board RAM. Lastly, you specify using jumper area H whether one of more of the last three 1K blocks is to be returned for use by other boards. (Normally you specify that all three of them are returned.)

Two typical configurations of Block B are shown in figures 2 and 3. Figure 2 is the standard - the one for which the board is pre-wired. Since in the pre-wired version, block B begins at C000H, Figure 2 shows that the standard address for scratch-pad RAM is DC00H, and the standard address for the System's Monitor PROM is C000H. Figure 3 shows the result of putting the PROM in the upper 4K and specifying that the RAM occupy the second 1K portion.


ON-BOARD PROM
SCRATCHPAD
EPROM
VIDEO
VIDEO
RAM
DISK
B+2000H
B+1C00H
B+1800H
B+1400H
B+1000H
B+0C00H
B+0800H
B+0400H
Starting Address B
FIGURE 3
RAM
DISK
VIDEO
VIDEO
EPROM
SCRATCHPAD
ON-BOARD PROM
FIGURE 2

2.6 BLOCK SELECT ADDRESSING

Jumper areas: E & F



Jumper names: A13, A13, A14, A14, A15, A15 = address lines
BA1, BA2, BA3 = block B address pads
BB1, BB2, BB3 = block A address pads

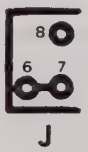
NOTE: The second letter in the block B address pads is "A", while the second letter in the block A address pads is "B". This occurs because historically, the pads were named before it was decided to manufacture the board with the "block swap" jumpers in area G reversed.

Function: Address lines A13, A14, A15 form the most significant bits of the address from the CPU. These three bits can select any of 8 possible 8K blocks of memory in a 64K memory space. See table 1.

Options: Table 2 tells you what jumpers to connect to specify any particular 8K block starting address.

2.7 PROM/SCRATCHPAD MEMORY INVERT

Jumper area: J



Function: The pre-wired connection specifies that the low order 4k bytes of block B consists of PROM. This jumper area is used to reverse this, putting the PROM at the high end of block B.

Options: If the PROM is to occupy the high order addresses of this block cut the jumper from 6 to 7 and tie 6 to 8.

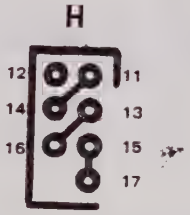
2.8 RAM MEMORY ADDRESS SELECT IN BLOCK B

Jumper area: I

Function: These jumpers allow the user to selectively determine where the RAM addresses are to be located. With the board jumpered as manufactured, the 1K of RAM occupies the top-most 1K of addresses of the 4K scratchpad memory block.

Options: If you wish to alter the factory supplied connections, the following procedure is recommended: Cut the jumper from 18 to RAC. Then, determine the desired address for the 1K RAM from Table 3 and connect a jumper as specified. The third part of Table 3 is not relevent to this jumper area.

2.9 DISABLE 3K OF ADDRESS SPACE IN BLOCK B



Jumper area: H

Function: These jumpers allow the user to selectively determine which 3 of 4 1K blocks of memory are returned for use by other boards. These jumpers are selected in conjunction with the RAM memory address jumper in area I, so that together, all 4K of the non-PROM (scratchpad) address space in block B are accounted for. The factory supplied connections complement the factory supplied RAM address jumper, so that the bottom 3K of the scratchpad memory is allocated for use by other boards.

Options: If it is desired to alter the factory supplied connections, the following procedure is recommended: Verify the RAM memory address selected previously. Then, refer to Table 3 to find the RAM address selected, and connect jumpers as specified in the third part of the table.

2.10 POWER-ON/RESET JUMP - DESCRIPTION

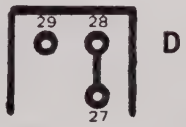
A power on/reset jump feature is also provided on this board. When the $\overline{\text{POC}}$ or $\overline{\text{PRESET}}$ (your choice of which, by jumper selection) line is low, the instruction stored in the first address of block A or B (determined by the jumper in area G, as explained below) will be executed by the CPU, and a "phantom" signal will be issued by the board on bus line 67 which disables other system memory boards.

After this initial instruction execution, the other memory boards will be re-enabled. However, if the instruction is a jump to the next instruction in the same block, then control will have been effectively transfered to that block on the PROM/RAM III board. Therefore, the second instruction should be the beginning of a system initialization routine followed by a systems executive. This is always the case in standard Vector Graphic computers.

Two additional jumper areas are provided, one to disconnect the phantom signal if it is not desired, and the other to disconnect the jump to the on-board PROM if this is not desired. These options give you maximum control over use of the board.

2.11 USE PRESET OR POC FOR POWER-ON/RESET JUMP

Jumper area: D



Function: In the factory version of the board, the $\overline{\text{POC}}$ signal is connected to the power-on/reset jump circuitry on the board. This is appropriate for standard Vector Graphic computers, because in these systems, both the RESET switch on the front panel and the initial power-on condition cause an active low pulse on the POC line, via circuitry on the Z80 board. If the CPU board used in your system does not have this feature, the $\overline{\text{PRESET}}$ signal can be connected to the power-on/reset circuitry by changing the jumper area D.

Options: To connect $\overline{\text{PRESET}}$ to the power-on/reset circuitry, cut the trace between 27 and 28 and tie 28 to 29.

2.12 PHANTOM GENERATED IF POWER-ON/RESET

Jumper area: C

C

Function: When 1 and 2 are tied together, the phantom signal is generated whenever a POC or PRESET signal is received. Phantom disables other system memory boards. The Z80 (and 8080) processor chip immediately executes the instruction at 0000H when the $\overline{\text{POC}}$ or $\overline{\text{PRESET}}$ signal appears on the bus, assuming the CPU board is so designed. With the other memory boards in the system disabled, the PROM/RAM III Board is free to supply the instruction for address 0000H.

Options: To disable the generation of the phantom signal, cut the jumper from 1 to 2.

2.13 JUMP TO PROM/RAM III BOARD IF POWER-ON/RESET

Jumper area: A

A

Function: When the $\overline{\text{POC}}$ or $\overline{\text{PRESET}}$ signal is received, a jumper in area A causes the board to respond to the address 0000H from the CPU. At your option, you may disable this feature, so that the PROM/RAM III board is NOT the board which responds to the address 0000H.

Options: To cause the board NOT to respond to address 0000H when $\overline{\text{POC}}$ or $\overline{\text{PRESET}}$ is received, cut the jumper from 3 to 4 and tie 4 to 5.

2.14 BLOCK SWAP

Jumper area: G

Function: With the board as manufactured, jumper area E is used to address block B, and jumper area F is used to address block A. Furthermore, if the power-on/reset jump feature is used, the jump will take place to the first address in block B.

Options: If you want to jump to block A instead, cut the jumpers from 20 to 21 and 25 to 26; tie 20 to 25 and 21 to 26. This change will also reverse the use of area E and F, so that area E is used to address block A, and area F is used to address block B.

2.15 DISABLE POWER-ON/RESET RESPONSE

To disable the power-on/reset response of the PROM/RAM III board entirely, disable both the generation of phantom and the jump to PROM/RAM III board. See Sections 2.12 and 2.13.

2.16 MWRITE INPUT

Jumper area: B

Function: If this board is installed in a system without a front panel, or other source of MWRITE, an MWRITE signal can be generated on board both for use on board and for feeding back to the bus as a fully buffered S-100 signal. This is not needed in Vector Graphic systems shipped after April 9, 1979, because the Z-80 boards in these systems now generate MWRITE.

Options: If the board is installed in a system without a source of MWRITE, add a jumper from 9 to 10.

2.17 WAIT STATE GENERATION

Jumper area: K

Function: The PRDY signal may be jumpered to the WAIT input in order to create one wait state each time the board is addressed. This is necessary when using memory slower than about 300 ns. in a 4 MHz (Z-80) system. PRDY is not connected to WAIT on the PROM/RAM III board as manufactured, because the Vector Graphic Z-80 board used in Vector Graphic systems generates the wait-state. You would want to generate the wait-state on the PROM/RAM III board if you are using memory faster than 300 ns. on other memory boards in the system, allowing you to disable the wait state that is built into the Vector Graphic Z-80 board (and some other manufacturers' Z-80 boards) yet continue to use a wait-state for the slower memory on the PROM RAM/III board.

For some Z-80 based CPU boards the WAIT output is not synchronized properly. If the WAIT is jumpered to the PRDY signal when such a Z-80 board is used, a possible oscillatory condition can arise on the PRDY and WAIT lines. Therefore, caution must be exercised in how this jumper is utilized. The Vector Graphic Z-80 board has a properly synchronized WAIT, so that with this Z-80 board, PRDY may be safely tied to WAIT, insuring reliable memory operation at high speeds.

Options: To tie PRDY to WAIT, jumper 22 to 23.

H = Hexidecimal D = Decimal

TABLE 1

A15	A14	A13	8K BLOCK (A or B) STARTING ADDRESS
0	0	0	0000H = 0000D
0	0	1	2000H = 8192D
0	1	0	4000H = 16384D
0	1	1	6000H = 24576D
1	0	0	8000H = 32768D
1	0	1	A000H = 40960D
1	1	0	C000H = 49152D
1	1	1	E000H = 57344D

x = Block A or B

If any Bx1, Bx2, Bx3 is tied to disable, that block of memory is disabled.

TABLE 2

DESIRED 8K BLOCK STARTING ADDRESS	CONNECT Bx1 to:	Bx2 to:	Bx3 to:
0000H	$\overline{A13}$	$\overline{A14}$	$\overline{A15}$
2000H	A13	$\overline{A14}$	$\overline{A15}$
4000H	$\overline{A13}$	A14	$\overline{A15}$
6000H	A13	A14	$\overline{A15}$
8000H	$\overline{A13}$	$\overline{A14}$	A15
A000H	A13	$\overline{A14}$	A15
C000H	$\overline{A13}$	A14	A15
E000H	A13	A14	A15

TABLE 3

ADDRESS OF 1K RAM WITH RESPECT TO THE STARTING ADDRESS OF THE 4K BLOCK	JUMPERS FOR RAM ADDRESS WITHIN 4K BLOCK	JUMPERS FOR BUS DISABLE
0000H	18 to RA0	15 to 16, 13 to 14, 11 to 12
0400H	16 to RA4	15 to 17, 13 to 14, 11 to 12
0800H	18 to RA8	15 to 17, 13 to 16, 11 to 12
0C00H	18 to RAC	11 to 14, 13 to 16, 15 to 17

2.18 PROGRAMMING A PROM

This board is accompanied by a program which allows you to program any 2704 or 2708 type EPROM. The listing of this program is found in Section 2.21, below. This same program is found on MDOS System Diskettes, version 8.4 and later, which accompany all Vector Graphic computers that are equipped with PROM/RAM III boards. The program exists on the disk as an immediately executable utility. The program is written in machine language and is not dependent on any operating system (except that it uses the Extended Systems Monitor in Vector Graphic systems for console I/O.) The utility (called "PROM") runs beginning at address 2B00 Hex and takes up less than 1K. If you want to run it elsewhere, or want to revise it, reassemble it as described in Section 2.20.

If you use an operating system other than MDOS, but you have the MDOS diskette, simply load the program under MDOS and copy it to a disk using the other system. To load it, just type PROM (return) followed by control-C, under MDOS. If you do not have the MDOS diskette, enter the program from the listing. Once it is loaded in memory, you can execute it from any executive, including the Extended Systems Monitor executive. The following explains the use of this program. If you are not using MDOS, then substitute the MDOS commands given here by those that are relevent to you.

1. Make sure the computer power is OFF. Wait at least five seconds before pulling out any circuit boards.

2. Unscrew and remove the cover of the computer.

3. Find the PROM/RAM III board. If you cannot easily reach PROM socket 11 with your hand, pull the board out.

4. Insert the PROM you wish to program in socket 11. This is the right-hand socket in the second row. Make sure to insert the PROM with its notch pointed to the top of the board. The PROM used MUST have been erased using ultraviolet erasing techniques, unless it is new. The computer cannot simply write over any previously used PROM, because programming involves turning logical 1's into 0's, but cannot go the other way. Erasing fills the PROM with 1's, like a new PROM.

5. Return the board to a slot which allows you to reach socket 11 without pulling the board out in the future, if possible.

6. Turn computer power ON.

7. If the system is not in the Extended Systems Monitor executive (indicated by the Monitor prompt *) then depress RESET on the computer front panel.

8. Mount the MDOS system diskette in drive 0 (the right-hand drive.) Then,

depress B on the keyboard. MDOS will take control, as indicated by the MDOS prompt >.

9. Load the object code to be stored on PROM into a free area of memory. Alternately, you may generate the desired code by assembling or compiling a higher level program.

10. Following the MDOS prompt >, type PROM (return). The PROM programming program will take control.

11. In response to the question "Starting from:", type the address in Hex of the first location you wish to program, within the block of memory assigned to PROM socket 11. Then press the RETURN key. Usually this starting address will be CC00. If programming less than the entire PROM, it can be any address between CC00 and CFF0. It must be an address ending in 0. If not, the machine will report "bad boundary address" and give you another chance. Letters must be in upper case. Do not tack on an H or any other symbol.

    CC00 is the starting address of PROM socket 11 if the board is left in factory-supplied format. If you enter an address outside the range CC00 to CFF0, the program will not accept it, and will report "out of range" and then give you another chance. If the addressing jumpers determining the location of socket 11 have been modified, you must modify the program to accept other addresses.

12. In reponse to the question "terminating at:", type the address in Hex of the last location you wish to program, within the block of memory assigned to PROM socket 11. Then press the RETURN key. Usually this terminating addresss will be CFFF for 2708 PROMs and CDFF for 2704 PROMs. If programming less than the entire PROM, it can be any address between CC0F and CFFF. It must be an address ending in F, and must be greater than the starting address. If not ending in F, the machine will report "bad boundary address" and then give you another chance.

    As with the starting address, if you enter an address outside the range CC0F to CFFF, the program will not accept it, and will report "Out of range" and then give you another chance. Therefore, if the addressing jumpers determining the location of socket 11 have been modified, you must modify the PROM programming program to accept other addresses.

    After entering the terminating address, the computer will either continue with the next question, or it will report "specified portion of PROM is not erased." This message means either that the terminating address is less than the starting address, or that the PROM is not new and was not properly erased. This message is strictly a warning, because in certain rare cases you may want to write over an unerased PROM. After the message, the system will continue with the next question. If you want to start over to correct your mistake, instead of continuing, then depress the ESC key. This takes the system back to the Monitor. To get back to MDOS from the Monitor, depress J. Then begin the program again at step 10, above.

13. In response to the question "Source address:", type the starting address in memory of the material you want to store on PROM. This can be any address in memory. Then press the RETURN key.

14. Slide the "programming" switch at the upper right-hand corner of the PROM/RAM III board to the LEFT.

15. Now, press the RETURN key again. This will begin programmming of the PROM. The computer must pass through the range of target addresses 256 times. A message will appear on the screen showing which pass the machine is currently on.

16. When programming is complete, one of two events will take place. If the computer detects no errors in comparing the programmed PROM without the original code, then the system will return to the MDOS executive or whichever other executive was used to call the programming program. If an error is discovered however, the screen will show the first address within the PROM at which a verification error was found. For example, if you forgot to slide the programming switch to the left, then, since the PROM will not have been programmed at all, the first address will be incorrect, so that the system will report an error at address CC00, or whatever was the starting address you had specified. After reporting the error, the system will return to the MDOS executive, so that you can start over.

17. When programming is complete, immediately slide the programming switch on the PROM/RAM III board to the RIGHT. Do not postpone this.

18. Remove the programmed PROM from socket 11. Alternately, you may use the PROM without removing it. For example, you may run a checksum of the PROM using the Extended System Monitor's Q command. To do this, depress control-Q or whichever other command your system uses to get to the Monitor executive. Then type Q CC00 CFFF. (The spaces will occur automatically.) The checksum, will appear immediately. (If PROM socket 11 has been readdressed, then use the appropriate addresses.) To return to MDOS from the Monitor, depress J.

2.19 WRITING A PROM PROGRAMMING PROGRAM

Although the PROM/RAM III board is supplied with a program for programming PROMs, this section explains the principles behind the program, for those wishing to write their own. The supplied program is listed in Section 2.21, for reference.

To program a 2708 or 2704 type EPROM, simply write the desired data to the locations assigned to PROM socket 11. The board hardware automatically interprets any writing of data to PROM socket 11 as an intent to program it. You do not have to program an entire PROM. You may program any part of it, down to blocks as short as 16 adjacent locations. Normally, you will program all 1K of a 2708 or all 512 bytes of a 2704. Write to all desired addresses in sequence. After finishing one such cycle, repeat it, using exactly the same data. You must repeat this cycle 256 times. In other words, you must write to each address 256 times, with a substantial delay between each time you write to each address. This delay is produced by the time taken to cycle through all the addresses, which is sufficiently long if 16 or more locations are programmed.

A good program has a comparison of the source and destination data, after programming the PROM is complete.

If your system has a dynamic memory board in it (such as all Vector Graphic systems shipped since about March 1, 1979), then there MUST be a delay loop after each byte is written to the PROM, so that the processor can refresh memory. The delay loop must execute at least 128 instructions each time it is accessed. You will find an example of this at the top of the fourth page in the listing in Section 2.21.

Before executing a programming procedure, you must slide the programming switch on the upper right-hand corner of the board TO THE LEFT. Then, put the PROM to be programmed into socket 11, which is the socket furthest to the right in the second row. After successfully programming it, slide the switch BACK. If you do not, you might accidently erase a PROM sitting in socket 11.

A PROM which you want to program must be either new or newly erased using the standard ultraviolet technique.

2.20 RE-ASSEMBLING THE PROM PROGRAMMING PROGRAM

The source code for the program is listed in Section 2.21 below. Enter the program using the MDOS editor LINEEDIT. You can assemble it wherever you like, although BC00 is not suggested because M.BASIC uses the very top of RAM for stack. The pre-assembled version on the diskette (under the name "PROM") is assembled to run at 2B00, at the beginning of the MDOS applications area. The program is less than 1K long.

You may modify PROM.S before you assemble it, by using the MDOS editor LINEEDIT. One modification which may be required are the addresses in the last two lines of PROM.S. You will have to change these if you change the jumpers on the PROM/RAM III board which assign the address of the on-board RAM. After entering and modifying the program, SAVE it on diskette under the name PROM.S. (Type NAME "PROM.S" (return) followed by SAVE (return) while in LINEEDIT.

To assemble PROM.S, use the ZSM assembler. With a diskette having both ZSM and PROM.S mounted in drive 0, and with MDOS in control, type ZSM "PROM.S" "PROM2" "E" (return). The assembler will ask where you want to run the program. Enter the address, for example 2B00H, that you want it to run at. Note that if the first character is a letter, it must be preceded by a 0 (zero), and the address must be followed by an H. The above ZSM statement will cause the program to be assembled with only errors printed. For other options possible with ZSM, see Section 4.5 of the User's Guide to Vector Graphic Systems Using MDOS.

After the assembly is complete, type TYPE "PROM2" 18 (return). This type will allow you to execute the program simply by typing PROM2 (return) while under MDOS.

If you want to put the PROM programming program on a PROM, in order to have a permanent PROM programming capability, first choose the memory location you want to give to this PROM, say E000, which is available on the PROM/RAM III board. Use this address when asked by the assembler where you want it to run at. Since there is no RAM at this address, you will have to load the assembled code into a different location before you can put it on a PROM. To do this change the type to 00 rather than 18, by typing TYPE "PROM2" 00 (return), after the assembly is complete. This will allow you to type LOAD "PROM2" 2B00 (return) after the MDOS prompt >, thus loading the code at RAM address 2B00, ready to be saved on a PROM.

2.21 PROM PROGRAMMING PROGRAM LISTING

```
Addr B1 B2 B3 B4 E Label         Opcd   Operand

0000               ****************************
0000               *                          *
0000               * Prom Programming Program *
0000               *        Version 1         *
0000               *   for the Prom/Ram III   *
0000               *                          *
0000               *      by Lance Lewis,     *
0000               *    Vector Graphic Inc.   *
0000               *       20-July-79         *
0000               *                          *
0000               ****************************
0000               *
0000               *
0000               * System equates
0000               *
0000    C003   =   INPUT           EQU    0C003H        ;character input (C00C on pre 3.0 monitors)
0000    C008   =   OUT             EQU    0C008H        ;video driver (C098 on pre 3.0 monitors)
0000               *
0000               * Definitions and Constants
0000               *
0000    CC00   =   PROM            EQU    0CC00H        ;prom address
0000    00FF   =   BLANK           EQU    0FFH          ;erased byte of prom
0000    0D0A   =   CRLF            EQU    0D0AH         ;carriage return linefeed
0000    000A   =   LF              EQU    0AH           ;linefeed
0000    000D   =   CR              EQU    0DH           ;carriage return
0000    0080   =   MSB             EQU    80H           ;most significant bit
0000               *
0000    2B00   =   ORIG            REQ    'Program to run at?'
0000                               ORG    ORIG          ;assemble here
2B00               *
2B00               * Here we go
2B00               *
2B00 E5                            PUSH   H             ;save HL
2B01 D5                            PUSH   D             ;save DE
2B02 C5                            PUSH   B             ;saev BC
2B03 F5                            PUSH   PSW           ;save AF
2B04 21 00 00                      LXI    H,0
2B07 39                            DAD    SP            ;HL=SP
2B08 22 03 2E                      SHLD   STACK         ;store it
2B0B 31 00 DD                      LXI    SP,0DD00H     ;reset stack pointer
2B0E               *
2B0E CD 43 2D                      CALL   PRINT         ;send message
2B11 0D 0A                         DD     CRLF
2B13 20 20 20 20                   DT     '      Vector Graphic'
2B17 20 20 56 65
2B1B 63 74 6F 72
2B1F 20 47 72 61
2B23 70 68 69 63
2B27 0D 0A                         DD     CRLF          ;print CRLF
2B29 20 20 50 72                   DT     '  Prom Programming System'
2B2D 6F 6D 20 50
2B31 72 6F 67 72
2B35 61 6D 6D 69
2B39 6E 67 20 53
2B3D 79 73 74 65
2B41 6D
```

```
Addr B1 B2 B3 B4 E Label          Opcd   Operand

2B42 0D 0A                        DD     CRLF
2B44               *
2B44 0A                           DB     LF              ;down a line
2B45 20 20 50 72                  DT     '  Program prom'
2B49 6F 67 72 61
2B4D 6D 20 70 72
2B51 6F 6D
2B53 0D 8A                        DD     CRLF+MSB        ;end of message
2B55               *
2B55 CD 43 2D     STARTADRS       CALL   PRINT           ;send message
2B58 20 20 53 74                  DTH    '  Starting from :'
2B5C 61 72 74 69
2B60 6E 67 20 66
2B64 72 6F 6D 20
2B68 BA
2B69 CD 4F 2D                     CALL   ADRS            ;get start address
2B6C DA 55 2B                     JC     STARTADRS       ;if invalid try again
2B6F CD 43 2D                     CALL   PRINT
2B72 0D 8A                        DD     CRLF+MSB        ;print CRLF
2B74 CD 24 2D                     CALL   RANGERR         ;check for error
2B77 DA 55 2B                     JC     STARTADRS       ;try again if error
2B7A CD 96 2D                     CALL   MOD             ;check boundery
2B7D DA 55 2B                     JC     STARTADRS       ;no good
2B80 EB                           XCHG                   ;DE=start adrs
2B81               *
2B81 CD 43 2D     ENDADRS         CALL   PRINT           ;send message
2B84 20 20 54 65                  DTH    '  Terminating at:'
2B88 72 6D 69 6E
2E8C 61 74 69 6E
2B90 67 20 61 74
2B94 BA
2B95 CD 4F 2D                     CALL   ADRS            ;get end address
2B98 DA 81 2B                     JC     ENDADRS         ;if invalid try again
2B9B CD 43 2D                     CALL   PRINT
2B9E 0D 8A                        DD     CRLF+MSB        ;carriage return linefeed
2BA0 CD 24 2D                     CALL   RANGERR         ;check for range error
2BA3 DA 81 2B                     JC     ENDADRS         ;try again if error
2BA6 23                           INX    H               ;compensate
2BA7 CD 96 2D                     CALL   MOD             ;check boundery
2BAA DA 81 2B                     JC     ENDADRS         ;no good
2BAD 44                           MOV    B,H             ;save end address
2BAE 4D                           MOV    C,L             ; in register pair BC
2BAF               *
2BAF 62                           MOV    H,D             ;save start address
2BB0 6B                           MOV    L,E             ; in register pair HL
2BB1 1A           TFFS            LDAX   D               ;get byte from prom
2BB2 FE FF                        CPI    BLANK           ;is it clear
2BB4 C2 B7 2D                     JNZ    BADPROM         ;print "bad prom"
2BB7 13                           INX    D               ;check next location
2BB8 CD F6 2C                     CALL   TEST            ;end of area
2BBB C2 B1 2B                     JNZ    TFFS            ;more to come
2BBE EB           RESTORE         XCHG                   ;restore registers
2BBF               *
2BBF CD 43 2D     SOURCEADRS      CALL   PRINT
2BC2 20 20 53 6F                  DTH    '  Source address:'
2BC6 75 72 63 65
```

```
Addr B1 B2 B3 B4 E Label          Opcd   Operand

2BCA 20 61 64 64
2BCE 72 65 73 73
2BD2 BA
2BD3 CD 4F 2D                     CALL   ADRS          ;get source address
2BD6 DA BF 2B                     JC     SOURCEADRS    ;if not valid try again
2BD9                *
2BD9 CD 43 2D                     CALL   PRINT         ;send message
2BDC 0D 0A                        DD     CRLF
2BDE                *
2BDE 00 0A                        DD     LF            ;format output
2BE0 20 20 54 75                  DT     '  Turn on the programming enable switch'
2BE4 72 6E 20 6F
2BE8 6E 20 74 68
2BEC 65 20 70 72
2BF0 6F 67 72 61
2BF4 6D 6D 69 6E
2BF8 67 20 65 6E
2BFC 61 62 6C 65
2C00 20 73 77 69
2C04 74 63 68
2C07 0D 0A                        DD     CRLF
2C09 20 20 48 69                  DTH    '  Hit return to continue?'
2C0D 74 20 72 65
2C11 74 75 72 6E
2C15 20 74 6F 20
2C19 63 6F 6E 74
2C1D 69 6E 75 65
2C21 BF
2C22                *
2C22 CD 03 C0       STAT          CALL   INPUT         ;check keyboard
2C25 CA 22 2C                     JZ     STAT          ;no character
2C28 FE 0D                        CPI    CR            ;is it a return
2C2A C2 22 2C                     JNZ    STAT          ;no try again
2C2D                *
2C2D CD 43 2D                     CALL   PRINT
2C30 0D 0A                        DD     CRLF
2C32 0A                           DB     LF
2C33 20 20 50 72                  DT     '  Programming in progress'
2C37 6F 67 72 61
2C3B 6D 6D 69 6E
2C3F 67 20 69 6E
2C43 20 70 72 6F
2C47 67 72 65 73
2C4B 73
2C4C 0D 0A                        DD     CRLF
2C4E 8A                           DB     LF+MSB        ;stop sending with linefeed
2C4F                *
2C4F AF                           XRA    A             ;zero
2C50 32 02 2E                     STA    PASS          ; pass counter
2C53                *
2C53 E5             SAVE          PUSH   H             ;save source address
2C54 D5                           PUSH   D             ;save it
2C55                *
2C55 7E             LOOP          MOV    A,M           ;get byte from source
2C56 12                           STAX   D             ;program it to destination
2C57                *
```

```
Addr B1 B2 B3 B4 E Label      Opcd   Operand

2C57 3E 64                    MVI    A,100          ;delay for dynamic memory
2C59 3D              DELAY    DCR    A              ;time up
2C5A C2 59 2C                 JNZ    DELAY          ;keep stalling
2C5D                 *
2C5D 23                       INX    H
2C5E 13                       INX    D              ;advance pointers
2C5F CD F6 2C                 CALL   TEST           ;end of block
2C62 C2 55 2C                 JNZ    LOOP           ;no keep going
2C65                 *
2C65 21 02 2E                 LXI    H,PASS         ;point to pass counter
2C68 34                       INR    M              ;256 passes
2C69 F5                       PUSH   PSW            ;save Z flag
2C6A C5                       PUSH   B              ;save end pointer
2C6B                 *
2C6B CD 43 2D                 CALL   PRINT          ;send message
2C6E 0D                       DB     CR
2C6F 20 20 50 61              DTH    '  Pass '
2C73 73 73 A0
2C76 7E                       MOV    A,M            ;get pass number
2C77 0E 00                    MVI    C,0            ;clear number of digits
2C79 06 FF           LDIV     MVI    B,-1           ;compensate for increment
2C7B 04              DIV      INR    B              ;increment quotient
2C7C D6 0A                    SUI    10             ;subtract 10 from dividend
2C7E D2 7B 2C                 JNC    DIV            ;can more be subtracted
2C81 C6 3A                    ADI    10+'0'         ;adjust remainder 0 to 9 ASCII
2C83 F5                       PUSH   PSW            ;add to list of remainders
2C84 0C                       INR    C              ;one more digit
2C85 78                       MOV    A,B            ;prepare for next division
2C86 B7                       ORA    A              ;was quotient zero
2C87 C2 79 2C                 JNZ    LDIV           ;more to come
2C8A F1              LOUT     POP    PSW            ;get a remainder
2C8B CD 08 C0                 CALL   OUT            ;print it
2C8E 0D                       DCR    C              ;out of digits
2C8F C2 8A 2C                 JNZ    LOUT           ;no then keep printing
2C92                 *
2C92 C1                       POP    B              ;restore end
2C93 F1                       POP    PSW            ;restore Z flag
2C94 D1                       POP    D              ;restore start address
2C95 E1                       POP    H              ;restore HL
2C96 C2 53 2C                 JNZ    SAVE           ;more passes to come
2C99                 *
2C99 1A              VERIFY   LDAX   D              ;get byte from prom
2C9A BE                       CMP    M              ;is it the same
2C9B C2 FC 2C                 JNZ    VERIFYERR      ;print error
2C9E 23                       INX    H
2C9F 13                       INX    D              ;advance pointers
2CA0 CD F6 2C                 CALL   TEST           ;end of block
2CA3 C2 99 2C                 JNZ    VERIFY         ;still more to test
2CA6                 *
2CA6 CD 43 2D                 CALL   PRINT
2CA9 0D                       DB     CR
2CAA 20 20 4E 6F              DT     '  No errors detected'
2CAE 20 65 72 72
2CB2 6F 72 73 20
2CB6 64 65 74 65
2CBA 63 74 65 64
```

```
Addr B1 B2 B3 B4 E Label          Opcd   Operand

2CBE 0D 8A                        DD     CRLF+MSB
2CC0               *
2CC0 CD 43 2D      END            CALL   PRINT
2CC3 20 20 54 75                  DT     '  Turn off the programming enable switch'
2CC7 72 6E 20 6F
2CCB 66 66 20 74
2CCF 68 65 20 70
2CD3 72 6F 67 72
2CD7 61 6D 6D 69
2CDB 6E 67 20 65
2CDF 6E 61 62 6C
2CE3 65 20 73 77
2CE7 69 74 63 68
2CEB 0D 8A                        DD     CRLF+MSB
2CED               *
2CED 2A 03 2E                     LHLD   STACK          ;retrieve SP
2CF0 F9                           SPHL                 ;move it back
2CF1 F1                           POP    PSW           ;restore registers
2CF2 C1                           POP    B
2CF3 D1                           POP    D
2CF4 E1                           POP    H
2CF5 C9                           RET                  ;bye-bye
2CF6               *
2CF6 78            TEST           MOV    A,B           ;get end byte
2CF7 BA                           CMP    D             ;same as start
2CF8 C0                           RNZ                  ;no then return
2CF9 79                           MOV    A,C
2CFA BB                           CMP    E             ;low half same
2CFB C9                           RET                  ;return with Z flag
2CFC               *
2CFC CD 43 2D      VERIFYERR      CALL   PRINT
2CFF 0D                           DB     CR
2D00 3F 20 76 65                  DTH    '? verification error at '
2D04 72 69 66 69
2D08 63 61 74 69
2D0C 6F 6E 20 65
2D10 72 72 6F 72
2D14 20 61 74 A0
2D18 EB                           XCHG
2D19 CD E8 2D                     CALL   HEX           ;print hex address
2D1C CD 43 2D                     CALL   PRINT
2D1F 0D 8A                        DD     CRLF+MSB
2D21 C3 C0 2C                     JMP    END
2D24               *
2D24 7C            RANGERR        MOV    A,H           ;get high address
2D25 FE CC                        CPI    PROM/256      ;valid address
2D27 DA 2E 2D                     JC     RANGEMES      ;no print message
2D2A FE D0                        CPI    PROM/256+4    ;valid address
2D2C 3F                           CMC                  ;compensate
2D2D D0                           RNC                  ;return with C in question
2D2E CD 43 2D      RANGEMES       CALL   PRINT
2D31 3F 20 6F 75                  DT     '? out of range'
2D35 74 20 6F 66
2D39 20 72 61 6E
2D3D 67 65
2D3F 0D 8A                        DD     CRLF+MSB
```

```
Addr B1 B2 B3 B4 E Label        Opcd   Operand

2D41 37                         STC                ;set error flag
2D42 C9                         RET
2D43              *
2D43 E3           PRINT         XTHL               ;save HL get SP
2D44 7E           LPRINT        MOV    A,M         ;get character
2D45 CD 08 C0                   CALL   OUT         ;print it
2D48 23                         INX    H           ;advance pointer
2D49 B7                         ORA    A           ;is MSB set
2D4A F2 44 2D                   JP     LPRINT      ;keep sending
2D4D E3                         XTHL               ;restore HL and adjusted SP
2D4E C9                         RET
2D4F              *
2D4F 21 00 00     ADRS          LXI    H,0         ;zero value
2D52 CD 03 C0     LADRS         CALL   INPUT       ;get character
2D55 CA 52 2D                   JZ     LADRS       ;is it there
2D58 CD 08 C0                   CALL   OUT         ;print it
2D5B FE 0D                      CPI    CR          ;was it a return
2D5D C8                         RZ                 ;thats it
2D5E D6 30                      SUI    '0'         ;reduce to hex
2D60 DA 7B 2D                   JC     INVAL       ;invalid entry
2D63 FE 0A                      CPI    10          ;alpha character
2D65 DA 72 2D                   JC     SAB
2D68 D6 07                      SUI    7           ;alpha bias
2D6A DA 7B 2D                   JC     INVAL       ;bad character
2D6D FE 10                      CPI    16          ;number out of range
2D6F D2 7B 2D                   JNC    INVAL
2D72 29           SAB           DAD    H           ;multiply address by 16
2D73 29                         DAD    H
2D74 29                         DAD    H
2D75 29                         DAD    H
2D76 85                         ADD    L           ;combine new value
2D77 6F                         MOV    L,A
2D78 C3 52 2D                   JMP    LADRS       ;keep going
2D7B              *
2D7B CD 43 2D     INVAL         CALL   PRINT
2D7E 0D 0A                      DD     CRLF
2D80 3F 20 69 6E                DT     '? invalid response'
2D84 76 61 6C 69
2D88 64 20 72 65
2D8C 73 70 6F 6E
2D90 73 65
2D92 0D 8A                      DD     CRLF+MSB
2D94 37                         STC                ;set error flag
2D95 C9                         RET
2D96              *
2D96 7D           MOD           MOV    A,L         ;get low byte
2D97 E6 0F                      ANI    0FH         ;mask low nibble
2D99 C8                         RZ                 ;if zero fine
2D9A CD 43 2D                   CALL   PRINT
2D9D 3F 20 62 61                DT     '? bad boundery address'
2DA1 64 20 62 6F
2DA5 75 6E 64 65
2DA9 72 79 20 61
2DAD 64 64 72 65
2DB1 73 73
2DB3 0D 8A                      DD     CRLF+MSB
```

```
Addr B1 B2 B3 B4 E Label        Opcd   Operand

2DB5 37                         STC                ;set error flag
2DB6 C9                         RET
2DB7                *
2DB7 CD 43 2D       BADPROM     CALL   PRINT
2DBA 3F 20 73 70                DT     '? specified portion of prom is not erased'
2DBE 65 63 69 66
2DC2 69 65 64 20
2DC6 70 6F 72 74
2DCA 69 6F 6E 20
2DCE 6F 66 20 70
2DD2 72 6F 6D 20
2DD6 69 73 20 6E
2DDA 6F 74 20 65
2DDE 72 61 73 65
2DE2 64
2DE3 0D 8A                      DD     CRLF+MSB
2DE5 C3 BE 2B                   JMP    RESTORE     ;continue and restore registers
2DE8                *
2DE8 7C             HEX         MOV    A,H         ;first the high byte
2DE9 CD ED 2D                   CALL   BYTE        ;print hex byte
2DEC 7D                         MOV    A,L         ;now the low byte
2DED                *
2DED CD F0 2D       BYTE        CALL   NIBBLE      ;print nibble
2DF0                *
2DF0 0F             NIBBLE      RRC                ;swap nibbles
2DF1 0F                         RRC
2DF2 0F                         RRC
2DF3 0F                         RRC
2DF4 F5                         PUSH   PSW         ;save A
2DF5 E6 0F                      ANI    0FH         ;mask high nibble
2DF7 C6 90                      ADI    90H         ;super short-cut
2DF9 27                         DAA                ;technique for converting
2DFA CE 40                      ACI    40H         ;binary to ASCII
2DFC 27                         DAA                ;ala NB
2DFD CD 08 C0                   CALL   OUT         ;print it
2E00 F1                         POP    PSW         ;restore A
2E01 C9                         RET
2E02                *
2E02                PASS        DS     1
2E03                STACK       DS     1
```

III. THEORY OF OPERATION

3.1 ADDRESSING

Address input lines A0 to A9 are buffered in line receivers U13 and U14. The outputs of U13 and U14 are then connected to both the PROM and RAM memory address pins. Address input lines A10 to A15 are buffered in U12 before use on the board. Lines A10 to A12 are inverted by the buffers and used as inputs to decoders U8 and U9. These three lines enable one of eight outputs on U8 or U9, depending on which decoder is enabled. Note that since A10 to A12 are inverted, the decoding sequence is reversed. When A10 to A12 are all "0", the number 7 output of the enabled decoder is selected. Each of the eight outputs from each decoder is used to enable a specific 2708 PROM or the 1K block of on-board RAM, or one of the three 1K segments which are not used on this board.

Address input lines A13 to A15 are used to enable one or the other decoder. Jumper Areas E and F determine which specific 8K block of memory corresponds to each decoder. The decoders are enabled by the output of U18-13 and U10-6. (They are enabled when their D input is a logic low "0".) Which decoder is enabled by which line depends on the jumpering in Area G. Jumper Area G can be used to switch the memory blocks thus assigned to each decoder.

Inversion of the on-board PROM and scratchpad memory address within block B may be accomplished by changing the jumper in Area J. This jumper determines whether or not the A12 address line is inverted by U11-4 before being used by decoder U9.

Selection of which 1K segment of the memory space will be assigned to the on-board RAM and which three 1K segments will be returned for use by other boards is handled by U9 outputs pins 1, 2, 3, 4, gate U10-12 and jumpers in Areas I and H. Any time an input to gate U10-12 goes low, this board is inhibited from putting data on the DI bus by forcing the DI line drivers to the high impedence state. Therefore, the three outputs of U9 which are connected to the inputs to U10-12 cause output from this board to be inhibited when one of the corresponding addresses appear on the address bus. Likewise, whichever U9 output is tied to the CE input to the RAM will enable the on-board RAM when that address appears.

3.2 DATA INPUT/OUTPUT

The DO lines from the S-100 bus contain data from the CPU to the memory. RAM is contained in two 2114 chips (U1 and U2). U1 contains the low four data bits in each location and U2 the high four bits. Thus DO0 to DO3 are tied to the data pins of U1 and DO4 to DO7 to the data pins of U2. These data bus lines are also tied in parallel to the eight data lines of each 1K byte PROM chip.

Data outputs from the RAM and PROM are connected to the input of a tri-state line driver U16 or U17. This parallel bussing of outputs from the memory chips is possible since all data outputs on the chips are tri-state.

3.3 CONTROL SIGNALS

U15 buffers the data lines inputting to the board. This buffer is enabled so long as U5-10 is low, which is true if U4-11 is high, which is true if either the on-board RAM is being written to or if PROM socket 11 is being written to. This logic is accomplished as follows. U4-6 is the NAND of MWRITE and the inverted (active high at U5-4) chip select for PROM socket 11, so that U4-6 is low if both PROM socket 11 is selected and MWRITE is active. U20-6 is the NAND of MWRITE and the inverted RAM chip select (active high at U5-13) so that U20-6 is low if both RAM is selected and MWRITE is active. Since U4-11 is the NAND of U4-6 and U20-6, U4-11 will be high if either U4-6 or U20-6 is low.

Writing of data into the RAM is controlled by MWRITE. Depending on the jumper in Area B, MWRITE can be taken from the bus (if a front panel is used or if there is another source of MWRITE in the system), or it can be generated from SOUT and $\overline{\text{PWR}}$ on this board. To generate MWRITE on the board, when SOUT and $\overline{\text{PWR}}$ are both low, U18-10 is high. This signal is buffered at U14-9 and is available both to the bus and the board as MWRITE. MWRITE is NANDED with the RAM chip select (inverted to active high at U5-13), giving the RD/$\overline{\text{WR}}$ signal for RAM. Why is this necessary, since the signals are combined within the 2114? It is not necessary in order to generate RD/$\overline{\text{WR}}$, but to enable the data bus input driver U15, as exlained above, we needed external active low signals specifically for writing to RAM and to PROM. Rather than putting another inverter on the board, the same signal is used for RD/$\overline{\text{WR}}$ to RAM. A low on RD/$\overline{\text{WR}}$ puts the chip in the write mode. Data on lines DO0 to DO7 will be written

into the RAMs, assuming the board has been addressed and the RAM selected by the chip enable from Area I.

When it is desired to read data from this board, the U19-6 must be low at the appropriate time, enabling the DI bus drivers U16 and U17. This is accomplished by generating the logic NAND function of numerous signals. When either block A or block B is selected, the output of U20-3 is high which is used as one input to U19-6. Another input to U19-6 is generated by SMEMR which indicates that a memory read is to be executed. SMEMR is inverted at U11-2, then gated through U18-1, before being connected to U19. To allow selective disabling of this board's data outputs for any of the three unused 1K memory blocks, the chosen chip select lines are connected to U10 pins 1, 2 and 13. So long as they are high (not active), then U10-12 is low. In combination with a low from U11-2 (inverted SMEMR), a high appears on U18-1, which goes to U19-1. Another input to U19-6 is from U18-4 which senses that both SOUT and SINP are low. The last input to U19-6 is PDBIN. When this signal is high it indicates that the DI lines are in the input mode. Therefore, when all four inputs are high, indicating on board memory can be read, U19-6 will go low, thus enabling the data output buffers U16 and U17.

The power on/reset jump feature is initiated by the $\overline{\text{POC}}$ or $\overline{\text{PRESET}}$ input (jumper option in Area D). Disabling of other system memory boards during the power on/reset jump is accomplished by the PHANTOM output from this board, assuming the other boards are so wired. The power on/reset feature is provided by an RS flip-flop in U20, with the $\overline{\text{POC}}$ or $\overline{\text{PRESET}}$ line from the bus connected to the set input (U20-9) of the flip-fop. The PHANTOM signal is generated by the U20-11 active low output, and the U20-8 active high output is used to set U18-13 low, thus enabling U8 or U9, depending on the jumper in Area G. Since the address on the bus will be 0000, this causes the processor to execute the first instruction in the enabled 8K block. If this instruction is a jump to the next instruction in the same block, then when that instruction is decoded causing a low at U10-8 and hence at U20-13, the flip-flip will reset and cancel the PHANTOM signal.

The PRDY signal can be tied to the WAIT input by jumpering Area K. If so, the PRDY driver is enabled whenever this board is addressed and the processor is not doing I/O (determined by U19 pins 9, 10, 12 and 13.) WAIT is low at this time, thus PRDY goes low, putting the processor in a wait state. This makes WAIT go high, so that when the next clock cycle occurs, PRDY goes high again. The result is a one-cycle WAIT state each time the board is addressed. Note there is an error in this logic: a wait state will be generated (if jumpered in Area K) so long as any part of blocks A or B are addressed, INCLUDING the 3K which are used by other boards. This other 3K may be a function such as video or disk controller, which should not have a wait state.

3.4 PROM PROGRAMMING

PROM socket 11 is used to program an EPROM. EPROMs are programmed as follows: With the desired data on the data inputs to the PROM and the desired low order address byte on the address lines to the PROM, chip select must be raised to 12V (rather than the usual 0 for reading and 5 for not-select.) Then after a delay of 10 micro-seconds, a 26V pulse on the chip's programming pin (pin 18) must occur for 400 micro-seconds. The CPU must be held in a wait state during this time, as well as an additional 1/2 micro-second. This will program one byte ONCE. Proper programming of 2708 EPROMs require that each byte be programmed 256 times, with a delay after each time. This is handled in software, which should program all the locations on the PROM once, and then repeat the cycle 256 times. Software does not have to send any special signal for programming a PROM, since hardware will interpret any memory write to the PROM as an intent to program it. Unintential writing to the PROM will thus cause programming if the 26V supply is accidently left on.

U3 contains two one-shots which are used to generate the timing for the programming pulse. Each of these one shots has different R and C values connected to it, creating different length pulses. A 10 micro-second active low pulse is generated at U3-4 and a 410 micro-second active high pulse is generated at U3-5. When these two are NANDED together at U4-3, the result is a 400 micro-second active low pulse following a 10 micro-second delay, as desired. This pulse begins when PSYNC (bus line 76) and clock-1 (bus line 25) are NANDED at U4-8 and put into U3-1 and U3-9, and at the same time the PROM socket 11 chip select arrives at U3-2 and U3-10. They will only fire if it is not a memory read cycle, because U11-2 keeps the one-shots reset (via reset pins U3-3 and U3-11) if SMEMR is active.

The low-high transition of the 410 micro-seond pulse at U3-5 generates an active low on XRDY (bus line 3) by inverting it at U6-2, in order to put the CPU in a wait state. This stays low for 1/2 micro-second after the pulse is over because of an RC delay tied to U6-2.

The 400 micro-second pulse is converted to active open at U6-10 and U6-12. The program pulse of 26V is then generated by a 2N3643 transister, using a supply voltage from U7 and related circuitry. U7 is turned on by the sliding programming switch. This switch must ONLY be on when programming a PROM, because erroneous writing to that PROM will otherwise alter it when not desired.

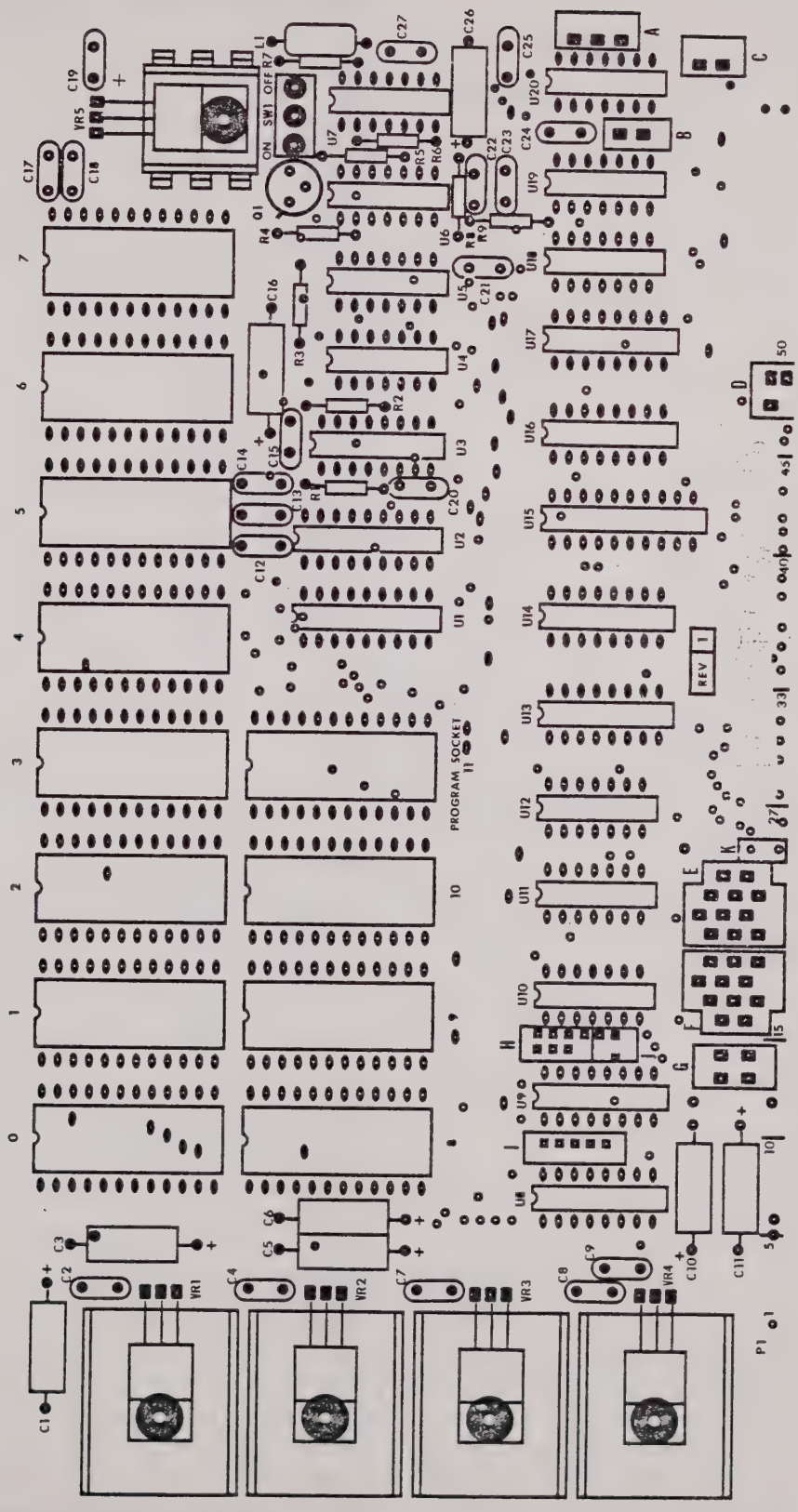
When the pulse is over and the wait line is released, the CPU is released to increment the address and program the next byte.

3.5 POWER SUPPLIES

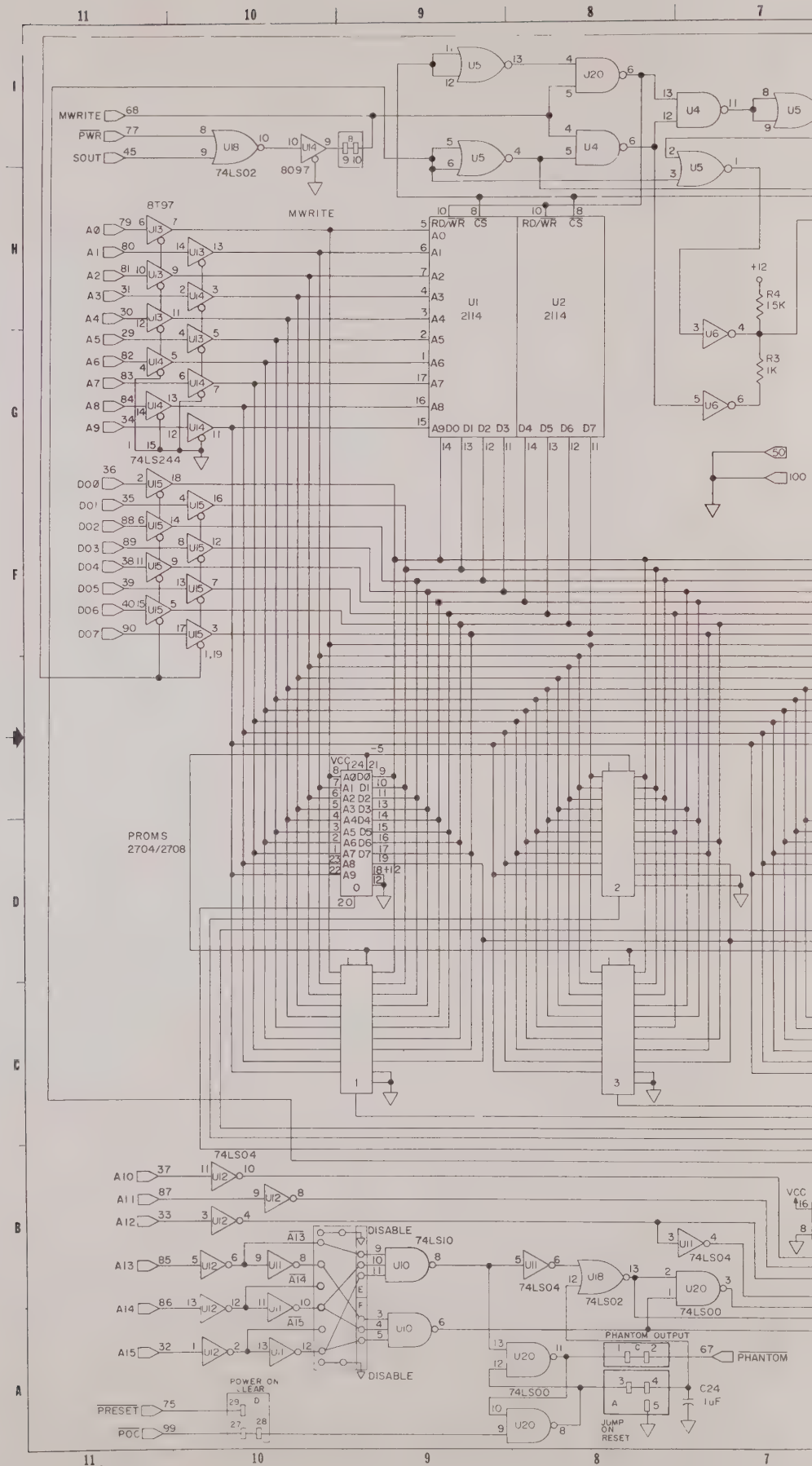
Power for this board is obtained from the unregulated +8V and plus or minus 18V supplies in the system.

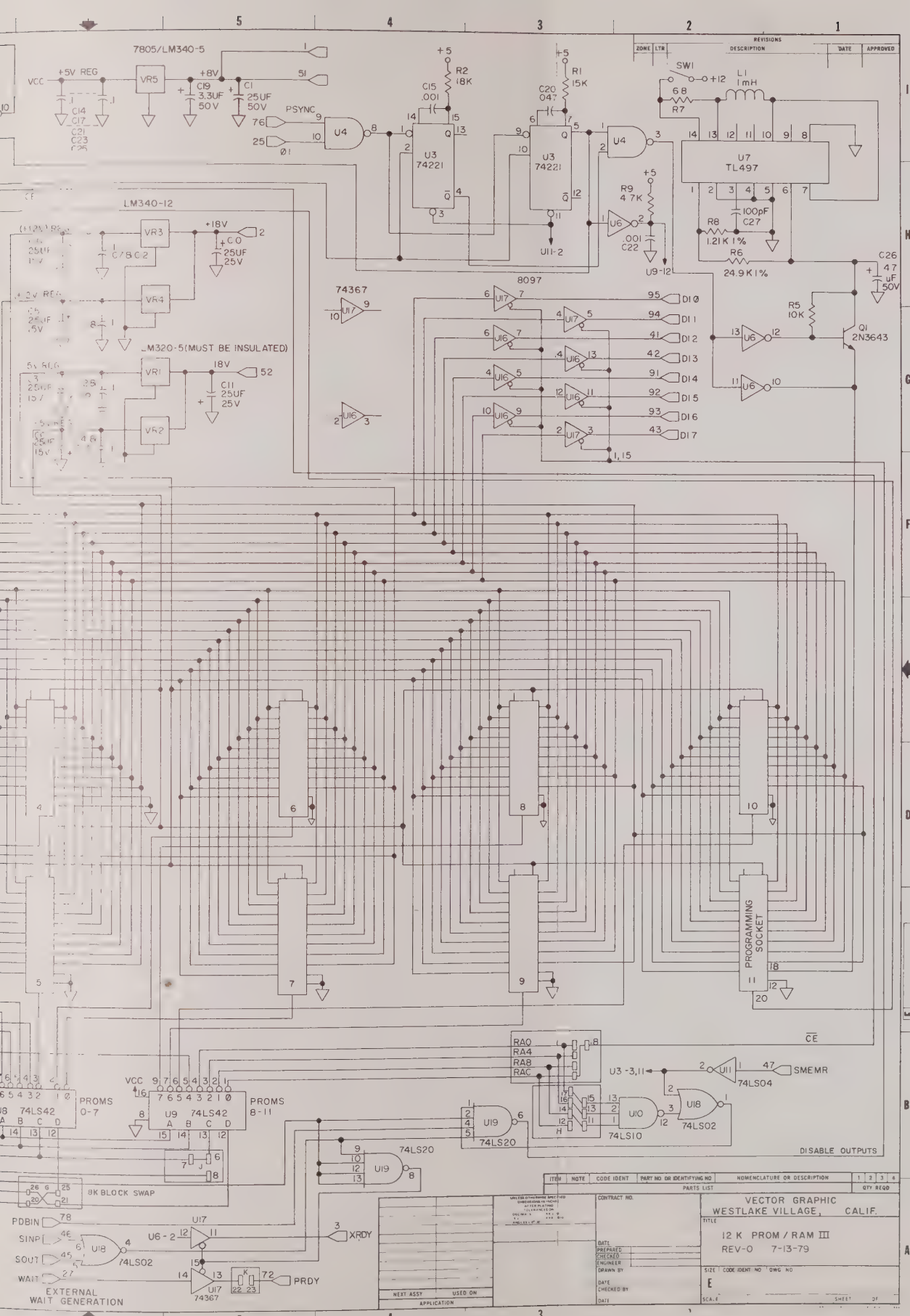
Regulation of the input voltage to the required -5V and +12V is obtained by the use of four three-terminal regulators. Dual regulators are used to insure ample supply current. The +5V supply is regulated by one regulator. Bypass filtering on all power lines is accomplished by multiple electrolytic capacitors for each supply voltage. This filtering insures stable noise free operation of the board. Capacitors are also used on each regulator input for high frequency bypassing and regulator stability.

The +26V programming supply is produced from the +12V regulated supply by a TL497 switching voltage regulator in a low-power step-up configuration, using a 1 mH coil.

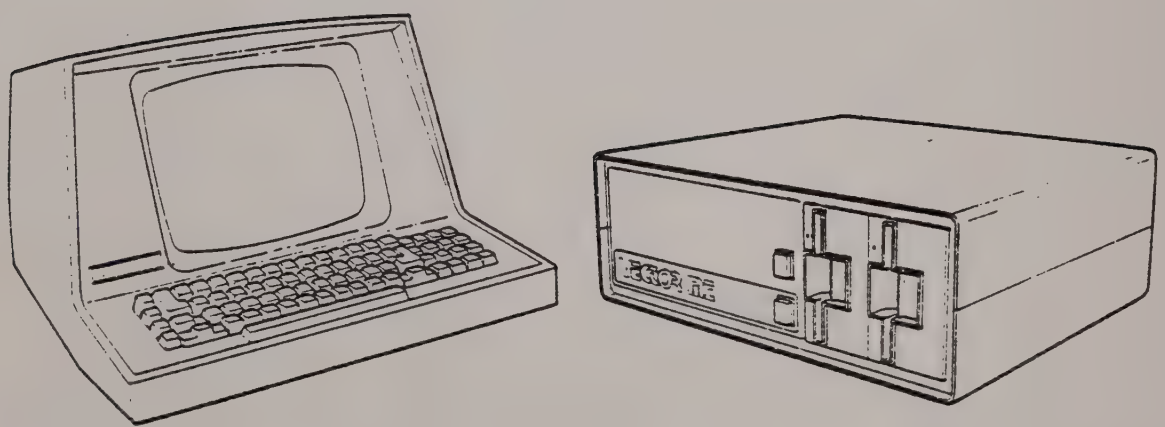

PROGRAM SOCKET
REV 1
SW1
ON
OFF
P1

SCHEMATIC ERRATA


MWRITE
PWR
SOUT
74LS02
8097
8T97
74LS244
MWRITE
RD/WR CS
U1 2114
U2 2114
+12
R4 15K
R3 1K
50
100
PROMS 2704/2708
74LS04
DISABLE
74LS10
74LS00
74LS02
PHANTOM OUTPUT
PHANTOM
C24 1uF
POWER ON CLEAR
PRESET
POC
JUMP ON RESET


7805/LM340-5
LM340-12
LM320-5(MUST BE INSULATED)
+5V REG
+8V
+18V
-5V REG
VR5
VR4
VR3
VR2
VR1
C19 3.3UF 50V
C1 25UF 50V
C0 25UF 25V
C11 25UF 25V
PSYNC
Ø1
U4
R2 18K
R1 15K
C15 .001
C20 047
U3 74221
SW1
L1 1mH
R7 68
U7 TL497
R8 1.21K 1%
C27 100pF
R6 24.9K 1%
R9 4.7K
C22 .001
U9-12
U11-2
C26 47 uF 50V
R5 10K
Q1 2N3643
74367
8097
DI0
DI1
DI2
DI3
DI4
DI5
DI6
DI7
PROGRAMMING SOCKET
RA0
RA4
RA8
RAC
U3-3,11
SMEMR
74LS04
74LS10
74LS02
74LS20
74LS42
PROMS 0-7
PROMS 8-11
DISABLE OUTPUTS
8K BLOCK SWAP
PDBIN
SINP
SOUT
WAIT
XRDY
PRDY
EXTERNAL WAIT GENERATION
REVISIONS
ZONE LTR
DESCRIPTION
DATE
APPROVED
PARTS LIST
ITEM
NOTE
CODE IDENT
PART NO OR IDENTIFYING NO
NOMENCLATURE OR DESCRIPTION
QTY REQD
CONTRACT NO.
VECTOR GRAPHIC
WESTLAKE VILLAGE, CALIF.
TITLE
12 K PROM / RAM III
REV-0 7-13-79
SIZE
CODE IDENT NO
DWG NO
E
SCALE
SHEET
DRAWN BY
CHECKED BY
ENGINEER
NEXT ASSY
USED ON
APPLICATION

Extended Systems Monitor



VECTOR GRAPHIC INC.

VECTOR 1 MONITOR - VERSION 2.1

THE VERSION 2.1 MONITOR IS A COMPACTLY WRITTEN PROGRAM DESIGNED TO FACILITATE MANIPULATION AND DISPLAY OF MEMORY DATA. THERE ARE 26 COMMANDS WHICH ARE ENTERED AS A SINGLE LETTER FOLLOWED BY UP TO FOUR HEXADECIMAL DATA FIELDS. AFTER EACH FIELD IS ENTERED, A SPACE IS AUTOMATICALLY OUTPUT AS A PROMPT. EITHER UPPER OR LOWER CASE ALPHA CHARACTERS CAN BE USED, BUT LOWER CASE CHARACTERS WILL BE CONVERTED TO UPPER CASE, AND ANY NON-HEX CHARACTERS WILL CAUSE THE COMMAND TO BE ABORTED. ALLOWABLE HEX CHARACTERS ARE 0-9, A-F. ADDRESS FIELDS ARE FOUR DIGITS LONG; OTHER FIELDS ARE TWO DIGITS LONG.

IF A SPACE IS TYPED AT ANY TIME DURING FIELD ENTRY, A DEFAULT VALUE OF ZERO IS ASSUMED FOR ALL LEADING ZEROES. THIS APPLIES TO AN ENTIRE FIELD AS WELL AS ONE THAT HAS BEEN PARTIALLY ENTERED, AND THE CURSOR WILL ADVANCE TO THE NEXT FIELD IF REQUIRED. FOR EXAMPLE, TYPING (SP) WILL HAVE THE SAME EFFECT AS TYPING 0000; TYPING 100(SP) WILL HAVE THE SAME EFFECT AS 0100.

THE HEXADECIMAL NUMBER SYSTEM MAY SEEM CONFUSING IF YOU ARE NOT FAMILIAR WITH IT, BUT IT HAS BECOME THE STANDARD OF THE MICROCOMPUTER FIELD AND IS CLEARLY THE BEST SYSTEM WITH 16 BIT ADDRESSES AND 8 BIT DATA. IT IS USUALLY NOT NECESSARY TO CONVERT BETWEEN NUMBER SYSTEMS, AS THIS IS USUALLY DONE BY SOFTWARE (I.E. ASSEMBLERS). REMEMBERING A FEW VALUES IN HEX SHOULD MAKE THINGS EASY:

HEX NUMBER	DECIMAL VALUE	JARGON	BINARY BITS
A	10		4
B	11		4
C	12		4
D	13		4
E	14		4
F	15		4
10	16		5
FF	255		8
100	256	1 PAGE	9
3FF	1,023		10
400	1,024	1K	11
FFF	4,095		12
1000	4,096	4K	13
4000	16,384	16K	15
8000	32,768	32K	16
FFFF	65,535	64K-1	16

THE FAMILIAR RULES OF ARITHMETIC WORK JUST THE SAME IN HEX AS IN DECIMAL:

```
      10          HEX (TRIVIAL)
40) 400
```

OR

```
      16          DECIMAL (MORE DIFFICULT)
64) 1024
    64
   ---
    384
    384
   ---
      0
```

SINCE MOST OF THE NUMBERS DEALT WITH ARE POWERS OF TWO, HEX IS EASIER.

COMMAND FORMAT

*

A <ADR1> <ADR2> - ASCII DUMP

MEMORY CONTENTS FROM ADR1 THROUGH ADR2 WILL BE DISPLAYED AS ASCII CHARACTERS, I.E. THE MOST SIGNIFICANT BIT OF EACH 8-BIT BYTE WILL BE IGNORED, AND VALUES LESS THAN 20 WHICH REPRESENT NON-PRINTING CONTROL CHARACTERS WILL BE PRINTED AS SPACES. THIS COMMAND IS USEFUL FOR EXAMINING FILES SUCH AS THOSE CREATED BY ESP OR THE MEMORITE WORD PROCESSOR. ASCII STRINGS IMBEDDED IN OBJECT CODE ARE EASY TO RECOGNIZE. THE DISPLAY MAY BE TEMPORARILY HALTED WITH THE SPACE BAR AND CONTINUED WITH ANOTHER SPACE. A CONTROL X WILL ABORT THE COMMAND.

*

B - JUMP TO BOOTSTRAP LOADER

TYPING THIS COMMAND WILL CAUSE A JUMP TO EITHER THE NORTH STAR DISK BOOTSTRAP LOADER PROM AT THE STANDARD LOCATION OF E900H IF IT IS RESIDENT IN THE SYSTEM OR TO THE MICROPOLIS BOOT (EXCEPTION: OPTION M, WHICH JUMPS TO BASIC AT 2A04). DO NOT USE THIS COMMAND UNLESS THE SYSTEM IS EQUIPPED WITH A DISK. THESE JUMP LOCATIONS ARE SUBJECT TO CHANGE IN FUTURE RELEASES OF THE MONITOR TO CONFORM WITH VECTOR GRAPHIC PRODUCTS.

*

C <ADR1> <ADR2> <ADR3> - COMPARE BLOCKS

A BYTE BY BYTE COMPARISON WILL BE MADE BETWEEN THE BLOCK OF MEMORY DATA STARTING AT ADR1 AND ENDING AT ADR2 AND A BLOCK OF IDENTICAL LENGTH STARTING AT ADR3. THE DIFFERENCES WILL BE PRINTED OUT WITH THE ADDRESS, THE BYTE IN THE FIRST BLOCK AND THE BYTE IN THE SECOND BLOCK. THIS COMMAND IS USEFUL TO COMPARE TWO VERSIONS OF A PROGRAM OR TO VERIFY THAT PROMS HAVE BEEN PROGRAMMED CORRECTLY.

*

D <ADR1> <ADR2> - DUMP IN HEX

MEMORY CONTENTS FROM ADR1 THROUGH ADR2 WILL BE DISPLAYED AS PAIRS OF HEXADECIMAL CHARACTERS. THE LEFT CHARACTER IN EACH PAIR REPRESENTS THE FOUR MOST SIGNIFICANT BITS OF THE MEMORY LOCATION. THE DISPLAY MAY BE HALTED AND INTERRUPTED AS DESCRIBED ABOVE IN THE 'A' COMMAND.

*

E - JUMP TO EXTENSION PROM

THIS COMMAND WILL CAUSE A JUMP TO LOCATION C400, WHICH IS THE NEXT AVAILABLE PROM LOCATION. THE PURPOSE OF THIS COMMAND IS TO PERMIT EASY EXTENSION OF THE MONITOR EITHER BY THE CUSTOMER OR BY THE ADDITION OF PROMS THAT WILL BE AVAILABLE FROM VECTOR GRAPHIC INC. IN THE FUTURE. THE MZOS DISK SYSTEM IS AVAILABLE AND REQUIRES A PROM AT THIS LOCATION.

*
F <ADR1> <ADR2> <BYTE1> <BYTE2> - FIND TWO BYTES

THE MEMORY RANGE FROM ADR1 THROUGH ADR2 WILL BE SEARCHED FOR THE PARTICULAR CODE COMBINATION BYTE 1 BYTE 2. THIS IS USEFUL FOR LOCATING PARTICULAR COMMANDS OR JUMP ADDRESSES. FOR EXAMPLE, IF YOU WISH TO CHANGE A CONTROL CHARACTER (SAY CONTROL D) IN A PROGRAM YOU MAY TRY FE 04, WHICH IS CPI 04 SINCE THIS IS A COMMON WAY OF TESTING INPUT CHARACTERS. IF YOU WISH TO FIND ALL LOCATIONS THAT CALL OR JUMP TO A PARTICULAR ADDRESS, SAY C700, THEN SEARCH FOR 00 C7. THERE IS NO GUARANTEE THAT EACH LOCATION DISPLAYED IS VALID OBJECT CODE - IT MAY BE PART OF A DATA TABLE, ASCII STRING, OR SECOND AND THIRD BYTES OF A THREE BYTE INSTRUCTION.

*
G <ADR1> - GO TO AND EXECUTE

THIS COMMAND WILL CAUSE A JUMP TO ADR1 TO EXECUTE A PROGRAM OR USER SUBROUTINE. AS WITH ALL MONITOR JUMP COMMANDS, THE ADDRESS CONTAINED ON THE STACK IS "START" (C00B) AND IF THE USER ROUTINE AT ADR1 ENDS IN "RET", PROGRAM EXECUTION WILL RETURN TO THE MONITOR. VIRTUALLY UNLIMITED STACK SPACE IS AVAILABLE (UP TO 1K), BUT OF COURSE, PUSHING MORE REGISTERS ON THE STACK THAN ARE POPPED WILL DEFEAT THE RETURN FEATURE WITH UNDESIRABLE EFFECTS.

*
H - JUMP TO 2A00

THIS COMMAND PERMITS EASY RETURN TO PROGRAMS IN THE USER APPLICATION AREA.

*
I <PORT> - INPUT FROM A PORT

EXECUTION OF THIS COMMAND WILL CAUSE THE CPU TO EXECUTE AN "IN PORT" INSTRUCTION AND THE ACCUMULATOR CONTENTS IMMEDIATELY FOLLOWING THIS TO BE DISPLAYED. THIS COMMAND IS USEFUL IN CHECKING OUT PERIPHERAL EQUIPMENT. ONLY THOSE PORTS USED BY THE TERMINAL, CASSETTE INTERFACE, ETC. WILL CONTAIN INTERESTING VALUES. ALL OTHERS WILL READ FF SINCE THE DATA BUS WILL BE FLOATING WHEN THE "IN" COMMAND IS EXECUTED.

*
J - JUMP TO LOADED DOS

THIS COMMAND PERMITS EASY RETURN TO THE LOADED NORTH STAR DOS OR MZOS AT 2028, OR IF NOT PRESENT THE JUMP WILL BE TO 04E7H, WHICH IS THE MICROPOLIS DOS ENTRY LOCATION (F000 IN OPTION M), AND IS SUBJECT TO CHANGE IN FUTURE RELEASES.

*
K - JUMP TO 0000

THIS COMMAND PERMITS EASY RETURN TO PROGRAMS WITH AN ENTRY POINT AT 0000, SUCH AS ESP AND SOME BASICS.

*

L <ADR1> <ADR2> - LOAD AND GO

TARBELL CASSETTE TAPES CAN BE LOADED WITH THIS COMMAND IF THE SYSTEM CONTAINS A TARBELL CASSETTE INTERFACE. DATA WILL BE LOADED INTO MEMORY FROM ADR1 THROUGH ADR2. AS THE DATA IS LOADED FROM TAPE, THE 8-BIT BYTE BY BYTE SUM IS COMPUTED (MOD 256 CHECKSUM). WHEN THE LAST MEMORY LOCATION HAS BEEN LOADED, THE NEXT BYTE ON THE TAPE IS COMPARED WITH THE COMPUTED CHECKSUM. THE COMPUTED CHECKSUM IS PRINTED AND IF BOTH VALUES AGREE, THE PROGRAM IS EXECUTED AT ADR1. IF THE VALUES DO NOT AGREE, AN "E" IS PRINTED AND THE MONITOR AWAITS A NEW COMMAND. A LOAD ERROR MAY OCCUR EITHER BECAUSE THE CASSETTE INTERFACE READS THE TAPE INCORRECTLY OR BECAUSE THE WRONG ADDRESSES HAVE BEEN ENTERED AND THE CHECKSUM IS NOT WHERE THE PROGRAM EXPECTS IT TO BE. NOTE THAT THE ADDRESS FORMAT IS DIFFERENT THAN THAT USED WITH THE ROUTINES SUPPLIED WITH THE CASSETTE INTERFACE, I.E. A BLOCK LENGTH OF 2000 (8K) STARTING AT LOCATION 3000 IS LOADED WITH L 3000 4FFF. THE ENDING LOCATION IS ONE LESS THAN THE BLOCK LENGTH ADDED TO THE STARTING ADDRESS.

*

M <ADR1> <ADR2> <ADR3> - MOVE MEMORY BLOCK

THE DATA CONTAINED IN MEMORY STARTING AT ADR1 AND ENDING AT ADR2 IS MOVED TO MEMORY LOCATIONS STARTING AT ADR3. THIS COMMAND IS USEFUL FOR MOVING A PROGRAM FROM A TEMPORARY STORAGE LOCATION TO ITS CORRECT ADDRESS. IF THERE IS AN OVERLAP OF THE TWO MEMORY AREAS, INTERESTING RESULTS ARE OBTAINED. FOR EXAMPLE, M 6000 7BFF 6400 WILL CAUSE THE BLOCK OF DATA FROM 6000 THROUGH 63FF TO BE REPEATED 8 TIMES FROM 6000 THROUGH 7FFF, SINCE BY THE TIME LOCATION 6400 IS READ, IT HAS BEEN OVERWRITTEN WITH DATA FROM 6000. THIS IS USEFUL FOR BANK PROGRAMMING OF PROMS, OR FOR CREATING REPEATING INSTRUCTION SEQUENCES FOR TEST PURPOSES.

*

N - NON-DESTRUCTIVE MEMORY TEST

MEMORY LOCATIONS STARTING AT 0000 ARE READ AND THE DATA TEMPORARILY STORED. THE MEMORY LOCATION IS THEN TESTED TO SEE IF 00 AND FF CAN BE WRITTEN AND READ CORRECTLY. THIS CONTINUES AFTER REWRITING THE ORIGINAL DATA UNTIL THE FIRST ERROR IS DETECTED, WHEREUPON THE ADDRESS IS DISPLAYED FOLLOWED BY THE DATA WRITTEN INTO MEMORY AND WHAT WAS READ FROM IT. THIS COMMAND IS MOST USEFUL FOR CHECKING HOW MUCH MEMORY A SYSTEM CONTAINS. FOR EXAMPLE, IF THE SYSTEM CONTAINS 16K OF MEMORY, 4000 00 FF SHOULD BE PRINTED, INDICATING THAT THERE IS NO MEMORY AT ADDRESS 4000. SINCE THE TEST IS NON-DESTRUCTIVE TO DATA IN MEMORY, IT CAN BE USED AT ANY TIME.

*

O <PORT> <DATA> - OUTPUT TO PORT

THE TWO HEX DIGITS "DATA" ARE LOADED INTO THE ACCUMULATOR AND THE INSTRUCTION "OUT PORT" IS EXECUTED. THIS COMMAND IS USEFUL FOR CHECKING OUT PERIPHERAL EQUIPMENT. FOR EXAMPLE, IF AN OPTION C I/O IS USED, O 02 41 WILL CAUSE AN "A" TO BE DISPLAYED ON THE TERMINAL SINCE 41 IS THE HEX ASCII CODE FOR "A".

*
P <ADR1> - PROGRAM MEMORY

THE CONTENTS OF MEMORY AT ADR1 ARE DISPLAYED FOLLOWED BY A DASH. IF THE SPACE BAR IS DEPRESSED, NO MODIFICATION WILL BE MADE TO THIS LOCATION AND THE NEXT LOCATION WILL BE DISPLAYED. TYPING TWO VALID HEX DIGITS WILL CAUSE THAT VALUE TO BE SUBSTITUTED INTO MEMORY AND THE NEXT LOCATION TO BE DISPLAYED. THE RETURN KEY WILL ONLY CAUSE A CRLF TO BE ECHOED. TO TERMINATE THE PROCEDURE, TYPE CONTROL X OR A NON-VALID HEX CHARACTER OTHER THAN SPACE OR RETURN.

*
Q <ADR1> <ADR2> - COMPUTE CHECKSUM

THE MOD 256 CHECKSUM OF MEMORY CONTENTS IN THE ADDRESS RANGE SPECIFIED IS COMPUTED AND DISPLAYED. THIS COMMAND IS USEFUL FOR CHECKING PROMS OR FILES TO SEE IF ANYTHING HAS CHANGED. ANY SOURCE FILE OR PROGRAM WRITTEN IN PURE CODE (IT DOES NOT WRITE ON ITSELF) WILL HAVE THE SAME CHECKSUM AS WHEN IT WAS LOADED. WHILE DEBUGGING ASSEMBLY LANGUAGE PROGRAMS, IT IS USEFUL TO BE ABLE TO VERIFY THAT A PROGRAM BEING DEBUGGED HAS NOT WRITTEN GARBAGE IN THE SOURCE FILE OR ASSEMBLER.

*
R <ADR1> <ADR2> - READ CASSETTE

THIS COMMAND IS SIMILAR TO THE "L" COMMAND, EXCEPT THAT IF THE LOAD IS GOOD IT PRINTS "R" AFTER THE CHECKSUM INSTEAD OF EXECUTING THE PROGRAM AT ADR1. THIS COMMAND IS USED IF MODIFICATIONS NEED TO BE MADE TO A PROGRAM, SUCH AS PATCHING THE I/O ROUTINES BEFORE IT IS RUN, OR IF THE GO ADDRESS IS NOT AT THE START OF THE PROGRAM.

*
S <ADR1> <ADR2> <BYTE> - SEARCH FOR SINGLE BYTE

THIS IS SIMILAR TO THE "F" COMMAND, EXCEPT THAT ONLY ONE BYTE IS SEARCHED FOR INSTEAD OF TWO. AN EXAMPLE OF THE USE OF THIS COMMAND IS TO DISPLAY ALL LOCATIONS IN A PROGRAM WHERE AN OUTPUT TO A PORT OCCURS (D3). THE ADDRESS OF EACH LOCATION WILL BE DISPLAYED FOLLOWED BY "D3" AND THE NEXT BYTE (THE PORT NUMBER).

*
T <ADR1> <ADR2> - TEST MEMORY

THIS IS AN EXTREMELY USEFUL COMMAND, ESPECIALLY WHEN FIRST SETTING UP A SYSTEM. THIS COMMAND PERMITS THOROUGH TESTING OF THE SYSTEM MEMORY. A PORTION OF A 64K BYTE PSEUDORANDOM NUMBER SEQUENCE IS WRITTEN INTO MEMORY FROM ADR1 THROUGH ADR2, AND THE EXACT SAME SEQUENCE IS REGENERATED FROM THE INITIAL POINT AND COMPARED WITH WHAT IS READ FROM MEMORY. IF ALL LOCATIONS COMPARE, ANOTHER PORTION OF THE SEQUENCE IS USED TO REPEAT THE TEST WHICH CONTINUES UNTIL IT IS INTERRUPTED. ANY MEMORY ERRORS ARE DISPLAYED WITH THE ADDRESS, WHAT WAS WRITTEN INTO MEMORY AND WHAT WAS READ FROM MEMORY, RESPECTIVELY. THIS INFORMATION IS ALL THAT IS NEEDED TO PINPOINT A MALFUNCTIONING MEMORY CHIP. REFER TO FURTHER INSTRUCTIONS

IN THE MEMORY BOARD INSTRUCTION MANUAL. THIS TEST IS QUITE EXHAUSTIVE IF USED FOR AT LEAST 10 CYCLES AND IS FAR SUPERIOR TO INCREMENTING OR COMPLEMENTING TESTS WHICH MAY NOT REVEAL ADDRESSING PROBLEMS. THE ONLY AREA OF SYSTEM MEMORY THAT CANNOT BE TESTED WITH THIS ROUTINE IS THE FEW BYTES REQUIRED FOR THE STACK AND VIDEO FLAGS IN THE VICINITY OF CFD0 ON THE 1702A PROM/RAM BOARD (DFD0 ON 2708 PROM/RAM).

*

U - TARBELL SYNC

THIS COMMAND CAUSES A CONTINUOUS STREAM OF SYNC CHARACTERS (E6) TO BE OUTPUT TO THE CASSETTE RECORDER IF THE SYSTEM CONTAINS A TARBELL INTERFACE. THIS IS USED TO RECORD A SYNC TAPE FOR USE IN ADJUSTING THE INTERFACE.

*

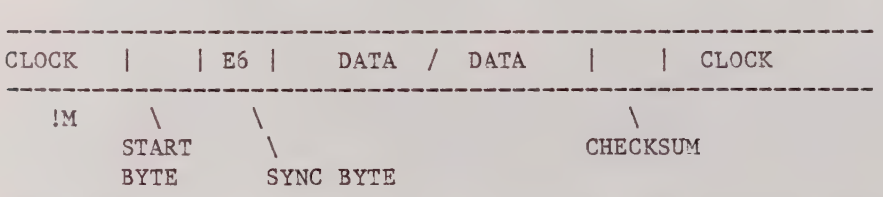
V <ADR1> <ADR2> - VERIFY TAPE

AFTER A TAPE HAS BEEN RECORDED WITH THE TARBELL CASSETTE INTERFACE, IT MAY BE VERIFIED USING THE "V" COMMAND. THE CHECKSUM IS COMPUTED FROM THE DATA READ BACK AND COMPARED WITH THE CHECKSUM ON THE TAPE. NO DATA IS TRANSFERRED TO MEMORY AND A BYTE BY BYTE COMPARISON IS NOT MADE WITH MEMORY. IF THE CHECKSUM AGREES, A "V" IS PRINTED AFTER IT. IF THE CHECKSUM PRINTED IS NOT THE SAME AS WHEN THE TAPE WAS RECORDED, BUT THE "V" IS PRINTED, YOU PROBABLY FORGOT TO PUSH THE RECORD BUTTON WHEN YOU STARTED TO RECORD, AND YOU ARE VERIFYING A PREVIOUS RECORDING.

*

W <ADR1> <ADR2> - WRITE CASSETTE TAPE

A BLOCK OF MEMORY DATA FROM ADR1 THROUGH ADR2 IS OUTPUT TO A TARBELL CASSETTE INTERFACE. THE FORMAT OF THE DATA IS AS FOLLOWS:



THE START BYTE ENSURES PRO NR CLOCK PHASE; THE SYNC BYTE TERMINATES HUNT MODE.

*

X <ADR1> <ADR2> <ADR3> - EXCHANGE MEMORY BLOCKS

A BLOCK OF MEMORY FROM ADR1 THROUGH ADR2 IS EXCHANGED WITH AN EQUAL LENGTH BLOCK STARTING AT ADR3. THIS COMMAND IS USEFUL IN COMPARING THE OPERATION OF TWO VERSIONS OF A PROGRAM, OR FOR RAPID SWITCHING OF PORTIONS OF A PROGRAM WITHOUT DESTROYING THE ORIGINAL. A LOADED BASIC PROGRAM CAN BE EXCHANGED WITH ANOTHER IF CARE IS USED TO INCLUDE THE STACK AREA (USUALLY BELOW THE TOP OF ALLOWED MEMORY).

*

Y <ADR1> <ADR2> - RELOCATING LOADER (VERSIONS WITHOUT VIDEO DRIVER)

A PROGRAM IS LOADED FROM CASSETTE STARTING AT ADR1 AND IS MODIFIED TO RUN AT THAT LOCATION. THIS IS A TWO PASS RELOCATING LOADER WHICH CAN BE USED WITH ANY STANDARD ASSEMBLER. THE DATA ON THE TAPE SHOULD BE GENERATED BY ASSEMBLING A PROGRAM TO RUN AT LOCATION 0000 AND RECORDING IT USING THE "W" COMMAND. IMMEDIATELY FOLLOWING THIS, RECORD THE SAME PROGRAM ASSEMBLED TO RUN AT LOCATION 0117. USING THE SAME BLOCK SIZE, THE PROGRAM CAN BE LOADED AT ANY ADDRESS USING THE "Y" COMMAND. OPTIONALLY, THE SECOND VERSION CAN BE ASSEMBLED AT XX00 WHERE XX DOES NOT EQUAL 00 BUT ADR1 IS THEN RESTRICTED TO START ON ANY PAGE BOUNDARY (YY00), WHICH IS A WISE THING FOR OTHER REASONS. THE ROUTINE LOADS IN THE FIRST VERSION STARTING AT ADR1 AND PRINTS "BAD LOAD" IF THE CHECKSUM IS NOT CORRECT. IT THEN READS THE SECOND VERSION ON THE TAPE AND MAKES A BYTE BY BYTE COMPARISON WITH MEMORY. BYTES THAT DIFFER BY 17 ARE RECOGNIZED AS LEAST SIGNIFICANT BYTES OF INTERNAL REFERENCES, AND THE LEAST SIGNIFICANT BYTE OF ADR1 IS ADDED TO MEMORY WITH ANY CARRY PROPAGATED TO THE NEXT BYTE WHICH WOULD BE MSB OF AN INTERNAL REFERENCE. BYTES THAT DIFFER BY ANYTHING OTHER THAN 17 ARE RECOGNIZED AS MOST SIGNIFICANT BYTES OF INTERNAL REFERENCES AND THE MOST SIGNIFICANT BYTE OF ADR1 IS ADDED TO MEMORY. IF THE SECOND VERSION LOADS SATISFACTORILY "GOOD LOAD" IS PRINTED. THE PROGRAM CAN NOW BE EXECUTED WITH THE "G" COMMAND.

THIS LOADER SHOULD NOT BE CONFUSED WITH "PROGRAM MOVERS" OPERATING ON OBJECT CODE. THEY HAVE NO WAY TO DISTINGUISH BETWEEN OBJECT CODE AND DATA FIELDS AND MAKE ERRORS WHICH MUST BE CORRECTED BY HAND.

AN EXAMPLE OF THE USE OF THE LOADER WITH A 1K PROGRAM IS AS FOLLOWS:

```
ESP:  AE 0000 4000      ASSEMBLE PROGRAM TO RUN AT 0 BUT LOAD OBJECT AT 4000
MON:   W 4000 43FF      RECORD OBJECT ON TAPE
ESP:  AE 0117 4000      ASSEMBLE TO RUN AT 0117 BUT LOAD OBJECT AT 4000
MON:   W 4000 43FF      RECORD OBJECT ON TAPE IMMEDIATELY AFTER ABOVE VERSION
MON:   Y 2000 23FF      LOAD AND RELOCATE PROGRAM AT 2000
```

*

Y - KEYBOARD ECHO (VERSIONS WITH VIDEO DRIVER)

THE MONITOR VERSIONS ON A WITH A VIDEO DRIVER HAVE THE RELOCATING LOADER REPLACED WITH THE VIDEO DRIVER, AND THE "Y" COMMAND JUMPS TO A KEYBOARD ECHO ROUTINE WHICH IS USEFUL FOR TEST PURPOSES. A CONTROL X WILL RETURN TO THE MONITOR.

*

Z <ADR1> <ADR2> <DATA> - ZERO OR FILL MEMORY

THE MEMORY BLOCK FROM ADR1 THROUGH ADR2 IS FILLED WITH THE BYTE "DATA". THIS IS USEFUL FOR SETTING MEMORY TO ZERO. THE END OF A FILE OR ASSEMBLED PROGRAM WILL STAND OUT MORE CLEARLY IF MEMORY IS FIRST ZEROED. FOR TEST PURPOSES, SINGLE INSTRUCTIONS CAN BE EXECUTED CONTINUOUSLY SO THAT BUS WAVEFORMS ARE MORE EASILY INTERPRETED. THIS IS DONE BY FILLING A BLOCK OF MEMORY WITH A REPEATED INSTRUCTION SEQUENCE WITH A JUMP TO THE START OF THE BLOCK SO THAT THE PROGRAM LOOPS CONTINUOUSLY.

OPTIONS, VERSIONS, ETC.

THE VERSION 2.1 MONITOR IS OFFERED IN NUMEROUS OPTIONS TO SUIT A RANGE OF AVAILABLE I/O BOARDS AND VIDEO DISPLAYS. IN ADDITION IT IS AVAILABLE ON TWO TYPES OF PROMS, THE 1702A FOR THE RESET AND GO PROM/RAM BOARD AND THE 2708 FOR THE 12K PROM/RAM BOARD. THE VARIOUS OPTIONS ARE LISTED BELOW, WITH ALL NECESSARY DATA TO DETERMINE HARDWARE COMPATIBILITY. TO DETERMINE THE OPTION TO USE WITH YOUR PARTICULAR HARDWARE, LOOK AT THE I/O ROUTINES SPECIFIED IN THE INSTRUCTION MANUAL. FOR INSTANCE, THE BITSTREAMER HAS THE FOLLOWING INPUT:

```
DB 03      RDCN    IN    STATUS
E6 02              ANI   RDA (LOOK AT RDA STATUS BIT)
CA                 JZ    RDCN (CONTINUE LOOPING IF IT IS NOT TRUE)
DB 02              IN    DATA (INPUT DATA)
```

FROM THIS WE SEE THAT THE STATUS PORT IS 3, THE RDA MASK IS 2, THE STATUS TRUE POLARITY IS HI AND THE DATA PORT IS 2. IF IT IS NECESSARY TO SET UP A PROGRAMMABLE USART AS IN THE OPTIONS B AND C IT IS DONE BY THE MONITOR INITIALIZATION ROUTINE. MANY I/O BOARDS CAN BE CONFIGURED FOR DIFFERENT I/O PORTS OR STATUS BITS. FOR EXAMPLE, THE BITSTREAMER NORMALLY USES THE C OPTION FOR SERIAL I/O, BUT IT CAN BE CONFIGURED TO WORK WITH A PARALLEL KEYBOARD INPUT AND OPTION EV OR AV. AT THE SAME TIME, THE SERIAL PORT CAN BE DRIVEN BY ANOTHER SOFTWARE I/O ROUTINE. IN INDICATING HARDWARE COMPATIBILITY, IT IS MEANT THAT A PARTICULAR BOARD CAN BE CONFIGURED TO WORK WITH THE PARTICULAR OPTION WITH PROPER JUMPERS AND/OR SWITCH SETTINGS.

THE I/O ROUTINES IN THE VERSION 2.1 MONITOR CAN BE USED AS THE MAIN SYSTEM I/O. THIS IS DONE SIMPLY BY PATCHING YOUR SOFTWARE TO CALL THE APPROPRIATE MONITOR ROUTINES. THIS MAKES SOFTWARE I/O INDEPENDENT AND EASILY INTERCHANGEABLE BETWEEN SYSTEMS. AN EXAMPLE OF HOW THIS IS DONE IS SHOWN BELOW:

```
INPUT ROUTINE:          INPT    CALL CNTLC (CODC)
                                JZ   INPT
                                RET  (RETURNS WITH CHAR INPUT IN A)
OUTPUT ROUTINE:         OUTPT   POP  PSW (CHAR TO BE OUTPUT WAS ON STACK)
                                JMP  PTCN (C098)
CONTROL X TEST:         CCNTL   CALL CNTLC (CODC)
                                RET  (RETURNS WITH ZERO FLAG SET IF NO INPUT,
                                     OR CHARACTER IN A.  JUMPS TO MONITOR IF
                                     CONTROL X INPUT)
```

NOTE THAT CONTROL X WILL BREAK TO THE MONITOR, WHICH PROVIDES A CONVENIENT WAY OF TRANSFERRING CONTROL FROM ANY EXECUTIVE SUCH AS THE DOS OR ESP TO THE MONITOR, BUT NECESSITATES THE USE OF ANOTHER CHARACTER (CONTROL C OR ESCAPE IS STANDARD) FOR A SINGLE LEVEL BREAK. THE ROUTINES ABOVE ARE MERELY GIVEN TO ILLUSTRATE HOW SIMPLE IT IS TO USE THE MONITOR I/O ROUTINES. MANY PROGRAMS REQUIRE ADDITIONAL INSTRUCTIONS TO MOVE THE CHARACTER TO BE OUTPUT INTO THE ACCUMULATOR, OR MAY REQUIRE DIFFERENT FLAG CONDITIONS OR ACCUMULATOR CONTENTS ON RETURN FROM THE INPUT AND CONTROL C ROUTINE, BUT THE VARIATIONS ARE EASILY IMPLEMENTED.

VIDEO DRIVER

The optional video driver in the monitor is a simplified version which emulates the elementary operation of a TTY. When the system is first turned on, the driver automatically clears the screen by testing an initialization flag in ram immediately above the stack area. The cursor is then written in the lower left corner of the screen. The control codes to which the driver responds are as follows:

Control D (04)-Clear the screen.

Control J or Linefeed (0A)-Scroll screen up one line and automatically return to the beginning of the next line.

Underline (5F)-Backspace and delete the character.

Control H or Backspace (08)-Same as above.

In addition to these functions, any other control character with a code less than 20 Hex will cause a fixed delay loop to be executed. The purpose of this is to allow BASIC to control the speed at which the screen scrolls by changing the number of NULL characters output at the end of a line.

There are three codes commonly used by various software for backspacing; 08 (backspace), 5F (underline), 7F (delete or rubout). By recognizing two of these, the video driver accomodates a majority of software without changes. One word of caution, however, is to ensure that the software does not merely echo the code without using it to backspace the internal buffer pointer.

MONITOR LISTINGS

Listings are provided for the option C and EV monitor versions. All other options are similar except for the I/O routines which are patched as shown in the table.

VECTOR GRAPHIC

MONITOR OPTIONS - VERSION 2.1

OPTION	# OF 1702A PROMS	# OF 2708 PROMS	INPUT - OUTPUT PARAMETERS					HARDWARE COMPATIBILITY
			STATUS PORT	RDA MASK	TRUE	DATA PORT	TBE MASK	
A	4	1	0	01	LO	1	80H	3P+S W/INVERTED STATUS MITS SIO REV 1
B	4	1	10H	01		11H	02H	MITS 2SIO
C	4	1	3	02H	HI	2	01H	BIT STREAMER, IMSAI SIO2
E	4	1	0	40H	HI	1	80H	3P+S Serial I/O
AV	4	1	0	01	LO	1	VIDEO	SERIAL INPUT, VDM-1 OUTPUT
EV	4	1	0	40H	HI	1	VIDEO	FLASHWRITER I/O (will also work with 3P+S input, VDM-1 out)

```
C000                    0010 *** VECTOR ONE MONITOR - VERSION 2.1
C000                    0020 * R. S. HARP 1/15/78
C000                    0030 * OPTION C (BITSTREAMER I/O)
C000                    0040 *
C000                    0050 * SYSTEM EQUATES
C000                    0060 CONS    EQU   3            CONS STATUS PRT
C000                    0070 COND    EQU   2            CONS DATA PORT
C000                    0080 TBE     EQU   1            TRANSMIT FLAG
C000                    0090 RDA     EQU   2            RECEIVE FLAG
C000                    0100 STPOL   EQU   0            STATUS POLARITY
C000                    0110 CASD    EQU   6FH          CASS DATA PORT
C000                    0120 CASC    EQU   6EH          CASS STAT PORT
C000                    0130 SPTR    EQU   0DFDOH       STACK POINTER
C000                    0170 *
C000                    0180 ********* COMMAND FORMAT *********
C000                    0190 * A SSSS FFFF ASCII DUMP
C000                    0200 * B JUMP TO BOOTSTRAP LOADER
C000                    0210 * C SSSS FFFF CCCC COMPARE BLOCKS
C000                    0220 * D SSSS FFFF DUMP IN HEX
C000                    0230 * E JUMP TO EXTENSION PROM
C000                    0240 * F SSSS FFFF DD DD TWO BYTE SEARCH
C000                    0250 * G SSSS GO TO AND EXECUTE
C000                    0255 * H JUMP TO 2A00
C000                    0260 * I PP INPUT FROM PORT
C000                    0270 * J JUMP TO DOS
C000                    0280 * K JUMP TO 0000
C000                    0290 * L SSSS FFFF LOAD AND GO
C000                    0300 * M SSSS FFFF DDDD MOVE BLOCK
C000                    0310 * N NON DESTRUCTIVE MEMORY TEST
C000                    0320 * O PP DD OUTPUT TO PORT
C000                    0330 * P LLLL PROGRAM MEMORY
C000                    0340 * Q SSSS FFFF COMPUTE CHECKSUM
C000                    0350 * R SSSS FFFF READ CASSETTE
C000                    0360 * S SSSS FFFF DD SEARCH FOR SINGLE BYTE
C000                    0370 * T SSSS FFFF TEST MEMORY
C000                    0380 * U GENERATES TARBELL SYNC
C000                    0390 * V SSSS FFFF VERIFY TAPE
C000                    0400 * W SSSS FFFF WRITE TAPE
C000                    0410 * X SSSS FFFF DDDD EXCHANGE BLOCK
C000                    0420 * Y SSSS FFFF RELOCATING LOADER
C000                    0430 * Z SSSS FFFF DD ZERO OR FILL MEMORY
C000                    0440 ************************************
C000                    0450 *
C000 C3 03 C0           0460 MONIT   JMP   INIT
C003 3E CE              0470 INIT    MVI   A,0CEH
C005 D3 03              0472         OUT   CONS
C007 3E 27              0474         MVI   A,27H
C009 D3 03              0476         OUT   CONS
C00B 31 D0 DF           0480 START   LXI   SP,SPTR
C00E CD A6 C0           0490         CALL  CRLF
C011 3E 2A              0500         MVI   A,'*'
C013 CD 98 C0           0510         CALL  PTCN
C016 CD BD C0           0520         CALL  RDCN          READ KEYBOARD
C019 E6 5F              0530         ANI   5FH           UPPER AND LOWER
C01B F5                 0540         PUSH  PSW           SAVE INPUT
C01C CD 96 C0           0550         CALL  SPCE
C01F F1                 0560         POP   PSW           SAVE INPUT
C020 21 0B C0           0570         LXI   H,START
C023 E5                 0580         PUSH  H
C024 FE 41              0590         CPI   'A'
C026 D8                 0600         RC                  TOO SMALL
C027 FE 5B              0610         CPI   05BH
C029 D0                 0620         RNC                 TOO LARGE
C02A 21 B5 C0           0630         LXI   H,CMDTB+7EH
C02D F5                 0640         PUSH  PSW
C02E 87                 0650         ADD   A
C02F 85                 0660         ADD   L
C030 6F                 0670         MOV   L,A
C031 5E                 0680         MOV   E,M
C032 23                 0690         INX   H
C033 56                 0700         MOV   D,M
C034 EB                 0710         XCHG
C035 F1                 0720         POP   PSW
C036 E9                 0730         PCHL                AWAY WE GO
C037                    0740 * COMMAND TABLE
C037 EE C1              0750 CMDTB   DW    DISP          A
C039 E1 C3              0760         DW    BOOT          B
C03B AC C2              0770         DW    COMPR         C
C03D EE C1              0780         DW    DISP          D
C03F 00 C4              0790         DW    MONIT+0400H   E
C041 C5 C2              0800         DW    SRCH          F
C043 6B C0              0810         DW    EXEC          G
C045 00 2A              0820         DW    02A00H        H
C047 21 C3              0830         DW    PINPT         I
C049 F4 C3              0840         DW    WARM          J
C04B 00 00              0850         DW    0             K
C04D 28 C1              0860         DW    CINR          L
C04F 71 C2              0870         DW    MOVEB         M
C051 93 C2              0880         DW    NDMT          N
C053 0A C3              0890         DW    POUTP         O
C055 23 C2              0900         DW    PGM           P
C057 ED C0              0910         DW    COUTR         Q
C059 28 C1              0920         DW    CINR          R
C05B C5 C2              0930         DW    SRCH          S
C05D 76 C1              0940         DW    TMEM          T
C05F 36 C3              0950         DW    SYNC          U
C061 28 C1              0960         DW    CINR          V
C063 ED C0              0970         DW    COUTR         W
C065 71 C2              0980         DW    MOVEB         X
C067 42 C3              0990         DW    RLODR         Y
C069 5D C2              1000         DW    ZEROM         Z
C06B                    1010 *
C06B                    1020 *** EXECUTE THE PROGRAM AT THE ADDRESS ***
C06B                    1030 *
C06B CD 70 C0           1040 EXEC    CALL  AHEX          READ ADD FROM KB
C06E EB                 1050         XCHG
C06F E9                 1060         PCHL                JUMP TO IT
C070                    1070 *
C070                    1080 *** CONVERT UP TO 4 HEX DIGITS TO BIN
C070                    1090 *
```

```
C070 0E 04
C072 21 00 00
C075 CD BD C0
C078 FE 20
C07A C3 D6 C3
C07D FE 3A
C07F D4 B0 C0
C082 29
C083 29
C084 29
C085 29
C086 D6 30
C088 FE 0A
C08A DA 8F C0
C08D D6 07
C08F 85
C090 6F
C091 0D
C092 C2 75 C0
C095 EB
C096 3E 20
C098 F5
C099 DB 03
C09B EE 00
C09D E6 01
C09F CA 99 C0
C0A2 F1
C0A3 D3 02
C0A5 C9
C0A6 3E 0D
C0A8 CD 98 C0
C0AB 3E 0A
C0AD C3 98 C0
C0B0 FE 41
C0B2 DA 0B C0
C0B5 E6 5F
C0B7 FE 47
C0B9 D2 0B C0
C0BC C9
C0BD
C0BD
C0BD
C0BD CD DC C0
C0C0 CA BD C0
C0C3 FE 60
C0C5 DA 98 C0
C0C8 E6 5F
C0CA C3 98 C0
C0CD CD DC C0
C0D0 FE 20
C0D2 C0
C0D3 CD DC C0
C0D6 FE 20
C0D8 C2 D3 C0
C0DB C9
C0DC DB 03
```

```
1100 AHEX   MVI   C,4          COUNT OF 4 DIGITS
1110 AHE0   LXI   H,0          16 BIT ZERO
1120 AHE1   CALL  RDCN
1130        CPI   20H
1140        JMP   PATCH
1150 PTHRT  CPI   ':'
1160        CNC   ALPH
1170        DAD   H            SHIFT 4 LEFT
1180        DAD   H
1190        DAD   H
1200        DAD   H
1210        SUI   48           ASCII BIAS
1220        CPI   10           DIGIT 0-10
1230        JC    ALF
1240        SUI   7            ALPHA BIAS
1250 ALF    ADD   L
1260        MOV   L,A
1270        DCR   C            4 DIGITS?
1280        JNZ   AHE1         KEEP READING
1290        XCHG
1300 SPCE   MVI   A,20H        PRINT SPACE
1310 PTCN   PUSH  PSW
1315 PTLOP  IN    CONS
1320        XRI   STPOL
1330        ANI   TBE
1340        JZ    PTLOP
1350        POP   PSW          THEN RECOVER A
1360        OUT   COND         AND PRINT IT
1370        RET                RETURN FROM PTCN
1380 CRLF   MVI   A,0DH        PRINT CR
1390        CALL  PTCN
1400        MVI   A,0AH
1410        JMP   PTCN
1420 ALPH   CPI   'A'
1430        JC    START
1440        ANI   5FH
1450        CPI   'G'
1460        JNC   START
1470        RET
1480 *
1490 *** READ FROM CONSOLE TO REG A ***
1500 *
1510 RDCN   CALL  CNTLC        READ KEYBOARD
1520        JZ    RDCN
1530        CPI   60H
1540        JC    PTCN
1542        ANI   5FH
1545        JMP   PTCN
1550 PAUSE  CALL  CNTLC
1560        CPI   20H
1570        RNZ
1580 PLOOP  CALL  CNTLC
1590        CPI   20H
1600        JNZ   PLOOP
1610        RET
1620 CNTLC  IN    CONS
```

```
C0DE EE 00
C0E0 E6 02
C0E2 C8
C0E3 DB 02
C0E5 E6 7F
C0E7 FE 18
C0E9 CA 0B C0
C0EC C9
C0ED
C0ED
C0ED F5
C0EE CD 03 C3
C0F1 06 00
C0F3 F1
C0F4 4F
C0F5 CD 13 C1
C0F8 3E E6
C0FA CD 13 C1
C0FD 7E
C0FE CD 13 C1
C101 80
C102 47
C103 CD 54 C2
C106 C2 FD C0
C109 78
C10A CD 13 C1
C10D CD D4 C1
C110 C3 0B C0
C113 F5
C114 79
C115 FE 51
C117 CA 24 C1
C11A CD CD C0
C11D DB 6E
C11F E6 20
C121 C2 1A C1
C124 F1
C125 D3 6F
C127 C9
C128
C128
C128
C128 F5
C129 3E 10
C12B D3 6E
C12D CD 03 C3
C130 F1
C131 E5
C132 F5
C133 06 00
C135 CD 69 C1
C138 4F
C139 F1
C13A F5
C13B FE 56
C13D 79
```

```
1630        XRI   STPOL
1640        ANI   RDA
1650        RZ
1660        IN    COND
1670        ANI   7FH
1680        CPI   18H          CNTL X
1690        JZ    START
1700        RET
1710 *** CASSETTE INTERFACE OUTPUT ROUTINE ***
1720 *
1730 COUTR  PUSH  PSW          SAVE CONTROL
1740        CALL  TAHEX
1750        MVI   B,0          START CHECKSUM = 0
1760        POP   PSW
1770        MOV   C,A
1780        CALL  COUT         START BYTE OUT
1790        MVI   A,0E6H       SEND SYNC BYTE
1800        CALL  COUT         TO CASSETTE
1810 COLOP  MOV   A,M          GET DATA FROM MEM
1820        CALL  COUT         SEND TO CASSETTE
1830        ADD   B            ADD TO CHECKSUM
1840        MOV   B,A
1850        CALL  BMP
1860        JNZ   COLOP        REPEAT LOOP
1870        MOV   A,B          GET CHECKSUM
1880        CALL  COUT         OUTPUT IT
1890        CALL  PT2          PRINT CHECKSUM
1900        JMP   START        GET ANOTH COMMND
1910 COUT   PUSH  PSW          SAVE A AND FLAGS
1920        MOV   A,C
1930        CPI   'Q'
1940        JZ    CHKSM
1950 CLOP   CALL  PAUSE
1960        IN    CASC         READ CASS STATUS
1970        ANI   20H          LOOK AT BIT 5
1980        JNZ   CLOP         TRY AGAIN?
1990 CHKSM  POP   PSW
2000        OUT   CASD         SEND DATA TO CASS
2010        RET   RETURN       FROM COUT
2020 *
2030 *** CASSETTE INPUT ROUTINE ***
2040 *
2050 CINR   PUSH  PSW          SAVE CONTROL CHAR
2060        MVI   A,10H        USE BIT 4 IN REG A
2070        OUT   CASC         TO RESET CASS INT
2080        CALL  TAHEX        READ ADDRESSES
2090        POP   PSW          GET CONTROL CHAR
2100        PUSH  H            SAVE START ADD
2110        PUSH  PSW          UNDER CONTROL CHAR
2120        MVI   B,0          SET CHECKSUM = 0
2130 CILOP  CALL  CIN          READ FM CONS
2140        MOV   C,A          SAVE IT IN REG C
2150        POP   PSW          GET CONTROL CHAR
2160        PUSH  PSW          SAVE IT BACK
2170        CPI   'V'          IS IT A V?
2180        MOV   A,C          GET BACK DATA BYTE
```

```
C13E CA 42 C1            JZ    CINO          IF C, DON'T STORE
C141 77                  MOV   M,A           IF NOT , STORE
C142 80                  ADD   B             ADD TO CHECKSUM
C143 47                  MOV   B,A
C144 CD 54 C2            CALL  BMP
C147 C2 35 C1            JNZ   CILOP         READ MORE
C14A CD 69 C1            CALL  CIN           READ LAST BYTE
C14D F5                  PUSH  PSW
C14E CD D4 C1            CALL  PT2           PRINT CHECKSUM
C151 CD 96 C0            CALL  SPCE          SPACE OVER
C154 F1                  POP   PSW
C155 B8                  CMP   B             COMP TO CHKSUM
C156 3E 45               MVI   A,'E'         PRINT E FOR ERROR
C158 C2 63 C1            JNZ   CERR          PRINT NOW IF ERROR
C15B F1                  POP   PSW           RECOVER CTL CHAR
C15C FE 4C               CPI   'L'           IF NOT L
C15E C2 63 C1            JNZ   CERR          DON'T EXECUTE
C161 E1                  POP   H             OTHERWISE, EXECJTE
C162 E9                  PCHL  AT            STARTING ADDRESS
C163 CD 98 C0    CERR    CALL  PTCN          PRINT V,E, OR R
C166 C3 0B C0            JMP   START
C169 CD CD C0    CIN     CALL  PAUSE
C16C DB 6E               IN    CASC          READ STATUS
C16E E6 10               ANI   10H           LOOK AT BIT 4
C170 C2 69 C1            JNZ   CIN           WAIT UNTIL LOW
C173 DB 6F               IN    CASD          READ DATA FM CASS
C175 C9                  RET   RETURN        FROM CIN
C176                     *
C176                     *** MEMORY TEST ROUTINE ***
C176                     *
C176 CD 03 C3    TMEM    CALL  TAHEX         READ ADDRESSES
C179 01 5A 5A            LXI   B,5A5AH       INI B,C
C17C CD A4 C1    CYCL    CALL  RNDM
C17F C5                  PUSH  B             KEEP ALL REGS
C180 E5                  PUSH  H
C181 D5                  PUSH  D
C182 CD A4 C1    TLOP    CALL  RNDM
C185 70                  MOV   M,B           WRITE IN MEM
C186 CD 54 C2            CALL  BMP
C189 C2 82 C1            JNZ   TLOP          REPEAT LOOP
C18C D1                  POP   D
C18D E1                  POP   H             RESTORE ORIG
C18E C1                  POP   B             VALUES OF
C18F E5                  PUSH  H
C190 D5                  PUSH  D
C191 CD A4 C1    RLOP    CALL  RNDM          GEN NEW SEQ
C194 7E                  MOV   A,M           READ MEM
C195 B8                  CMP   B             COMP MEM
C196 C4 C8 C1            CNZ   ERR           CALL ERROR ROUT
C199 CD 54 C2            CALL  BMP
C19C C2 91 C1            JNZ   RLOP
C19F D1                  POP   D
C1A0 E1                  POP   H
C1A1 C3 7C C1            JMP   CYCL
C1A4                     *** THIS ROUTINE GENERATES RANDOM NOS ***
C1A4 CD CD C0    RNDM    CALL  PAUSE
C1A7 78
C1A8 E6 B4
C1AA A7
C1AB EA AF C1
C1AE 37
C1AF 79
C1B0 17
C1B1 4F
C1B2 78
C1B3 17
C1B4 47
C1B5 C9
C1B6
C1B6
C1B6
C1B6 CD A6 C0
C1B9 CD CD C0
C1BC 7C
C1BD CD D4 C1
C1C0 7D
C1C1 CD D4 C1
C1C4 CD 96 C0
C1C7 C9
C1C8 F5
C1C9 CD B6 C1
C1CC 78
C1CD CD D4 C1
C1D0 CD 96 C0
C1D3 F1
C1D4 F5
C1D5 CD DC C1
C1D8 F1
C1D9 C3 E0 C1
C1DC 1F
C1DD 1F
C1DE 1F
C1DF 1F
C1E0 E6 0F
C1E2 C6 30
C1E4 FE 3A
C1E6 DA 98 C0
C1E9 C6 07
C1EB C3 98 C0
C1EE
C1EE
C1EE
C1EE 47
C1EF CD 03 C3
C1F2 0E 10
C1F4 CD B6 C1
C1F7 78
C1F8 FE 41
C1FA 7E
C1FB CA 0F C2
C1FE CD D4 C1
C201 CD 96 C0
                         MOV   A,B           LOOK AT B
                         ANI   0B4H          MASK BITS
                         ANA   A             CLEAR CY
                         JPE   PEVE          JUMP IF EVEN
                         STC
PEVE     MOV   A,C       LOOK AT C
         RAL             ROTATE CY IN
         MOV   C,A       RESTORE C
         MOV   A,B       LOOK AT B
         RAL             ROTATE CY IN
         MOV   B,A       RESTORE B
         RET             RETURN W NEW B,C
*
*** ERROR PRINT OUT ROUTINE
*
PTAD     CALL  CRLF      PRINT CR,LF
         CALL  PAUSE
         MOV   A,H       PRINT
         CALL  PT2       ASCII
         MOV   A,L       CODES
         CALL  PT2       FOR
         CALL  SPCE      ADDRESS
         RET
ERR      PUSH  PSW       SAVE ACC
         CALL  PTAD      PRINT ADD.
         MOV   A,B       DATA
         CALL  PT2       WRITTEN
         CALL  SPCE
         POP   PSW       DATA READ
PT2      PUSH  PSW
         CALL  BINH
         POP   PSW
         JMP   BINL
BINH     RAR
         RAR
         RAR
         RAR
BINL     ANI   0FH       LOW 4 BITS
         ADI   48        ASCII BIAS
         CPI   58        DIGIT 0-9
         JC    PTCN
         ADI   7         DIGIT A-F
         JMP   PTCN
*
*** DISPLAY MEMORY CONTENTS ***
*
DISP     MOV   B,A       SAVE CONTROL
         CALL  TAHEX     READ ADDRESSES
ENT1     MVI   C,16      LOC/LINE
         CALL  PTAD
LP2      MOV   A,B
         CPI   'A'       IS IT "A"?
         MOV   A,M
         JZ    ASCD      DUMP ASCII
         CALL  PT2       PRINT OUT
         CALL  SPCE
```

```
C204 CD 54 C2
C207 C8
C208 0D
C209 CA F2 C1
C20C C3 F7 C1
C20F E6 60
C211 C2 1A C2
C214 CD 96 C0
C217 C3 04 C2
C21A 7E
C21B E6 7F
C21D CD 98 C0
C220 C3 04 C2
C223
C223
C223
C223 CD 70 C0
C226 EB
C227 CD A6 C0
C22A 7E
C22B CD D4 C1
C22E 3E 2D
C230 CD 98 C0
C233 CD BD C0
C236 FE 20
C238 CA 50 C2
C23B FE 0D
C23D C2 46 C2
C240 CD A6 C0
C243 C3 33 C2
C246 EB
C247 21 00 00
C24A 0E 02
C24C CD 78 C0
C24F 73
C250 23
C251 C3 2A C2
C254
C254 7B
C255 95
C256 C2 5B C2
C259 7A
C25A 9C
C25B 23
C25C C9
C25D
C25D CD 03 C3
C260 E5
C261 0E 02
C263 CD 72 C0
C266 EB
C267 E3
C268 C1
C269 71
C26A CD 54 C2
C26D C8
```

```
3310 LP3    CALL  BMP
3320        RZ
3330        DCR   C
3340        JZ    ENT1          END OF LINE
3350        JMP   LP2           CONTINUE LOOP
3360 ASCD   ANI   60H           MASK FOR CONTROL
3370        JNZ   NCON
3380        CALL  SPCE
3390        JMP   LP3
3400 NCON   MOV   A,M
3410        ANI   7FH           MASK FOR ASCII
3420        CALL  PTCN
3430        JMP   LP3
6990 *
7000 *** PROGRAM MEMORY *****
7010 *
7020 PGM    CALL  AHEX          READ ADD.
7030        XCHG
7040        CALL  CRLF
7050 PGLP   MOV   A,M           READ MEMORY
7060        CALL  PT2           PRINT 2 DIG.
7070        MVI   A,'-'         LOAD DASH
7080        CALL  PTCN          PRINT DASH
7090 CRIG   CALL  RDCN
7100        CPI   20H           SPACE
7110        JZ    CON2          SKIP IF SPACE
7120        CPI   0DH
7130        JNZ   CON1          SKIP IF CR
7140        CALL  CRLF          PRINT CR,LF
7150        JMP   CRIG          BACK FO MO
7160 CON1   XCHG                H,L>D,E
7170        LXI   H,0           GET 16 BIT ZERO
7180        MVI   C,2           COUNT 2 DIG.
7190        CALL  AHE1+3        CONV TO HEX
7200        MOV   M,E           WRITE IN MEM
7210 CON2   INX   H             INC POINTER
7220        JMP   PGLP          KEEP GOING
7230 * COMPARE ADDRESSES AND INCREMENT H
7240 BMP    MOV   A,E
7250        SUB   L
7260        JNZ   GOON
7270        MOV   A,D
7280        SBB   H
7290 GOON   INX   H
7300        RET
7310 * ZERO OR FILL MEMORY WITH A CONSTANT
7320 ZEROM  CALL  TAHEX         READ ADDRESSES
7330        PUSH  H             SAVE H
7340        MVI   C,2           2 DIGITS
7350        CALL  AHE0          READ 2 DIGITS
7360        XCHG
7370        XTHL                RESTORE H,L
7380        POP   B
7390 ZLOOP  MOV   M,C           WRITE INTO MEM
7400        CALL  BMP           COMP ADD, INCREMENT H
7410        RZ                  RETURN IF DONE
```

```
C26E C3 69 C2
C271
C271 47
C272 CD 03 C3
C275 E5
C276 CD 70 C0
C279 EB
C27A E3
C27B 4E
C27C E3
C27D 78
C27E FE 4D
C280 CA 87 C2
C283 7E
C284 E3
C285 77
C286 E3
C287 71
C288 23
C289 E3
C28A CD 54 C2
C28D CA 0B C0
C290 C3 7B C2
C293
C293 21 00 00
C296 4E
C297 06 FF
C299 70
C29A 7E
C29B B8
C29C C2 C8 C1
C29F 06 00
C2A1 70
C2A2 7E
C2A3 B8
C2A4 C2 C8 C1
C2A7 71
C2A8 23
C2A9 C3 96 C2
C2AC
C2AC CD 03 C3
C2AF E5
C2B0 CD 70 C0
C2B3 EB
C2B4 7E
C2B5 23
C2B6 E3
C2B7 BE
C2B8 46
C2B9 C4 C8 C1
C2BC CD 54 C2
C2BF E3
C2C0 C2 B4 C2
C2C3 F1
C2C4 C9
C2C5
```

```
7420        JMP   ZLOOP           CONTINUE TIL DONE
7430 * MOVE A BLOCK OF MEMORY
7440 MOVEB  MOV   B,A
7450        CALL  TAHEX           READ ADDRESSES
7460        PUSH  H
7470        CALL  AHEX
7480        XCHG
7490        XTHL                  BACK TO NORMAL
7500 MLOOP  MOV   C,M
7510        XTHL
7520        MOV   A,B
7530        CPI   'M'
7540        JZ    NEXCH
7550        MOV   A,M
7560        XTHL
7570        MOV   M,A
7580        XTHL
7590 NEXCH  MOV   M,C
7600        INX   H
7610        XTHL
7620        CALL  BMP
7630        JZ    START
7640        JMP   MLOOP
7650 * NON DESTRUCTIVE MEMORY TEST
7660 NDMT   LXI   H,0             START AT ZERO
7670 NDLOP  MOV   C,M
7680        MVI   B,0FFH
7690        MOV   M,B
7700        MOV   A,M
7710        CMP   B
7720        JNZ   ERR             PRINT ERROR
7730        MVI   B,0
7740        MOV   M,B
7750        MOV   A,M
7760        CMP   B
7770        JNZ   ERR
7780        MOV   M,C
7790        INX   H
7800        JMP   NDLOP
7810 * COMPARE TWO BLOCKS OF MEMORY
7820 COMPR  CALL  TAHEX
7830        PUSH  H
7840        CALL  AHEX
7850        XCHG
7860 VMLOP  MOV   A,M
7870        INX   H
7880        XTHL
7890        CMP   M
7900        MOV   B,M
7910        CNZ   ERR
7920        CALL  BMP
7930        XTHL
7940        JNZ   VMLOP
7950        POP   PSW
7960        RET
7970 * SEARCH FOR SPECIFIC CODES
```

```
C2C5 F5          7980 SRCH   PUSH  PSW
C2C6 CD 03 C3    7990        CALL  TAHEX
C2C9 E5          8000        PUSH  H             SAVE H
C2CA 0E 02       8010        MVI   C,2           COUNT OF 2
C2CC CD 72 C0    8020        CALL  AHEO          READ 2 DIGITS
C2CF EB          8030        XCHG                H=CODE,D=F
C2D0 45          8040        MOV   B,L           PUT CODE IN B
C2D1 E1          8050        POP   H             RESTORE H
C2D2 F1          8060        POP   PSW
C2D3 FE 53       8070        CPI   'S'
C2D5 F5          8080        PUSH  PSW
C2D6 CA E2 C2    8090       JZ    CONT
C2D9 E5          8100        PUSH  H
C2DA 0E 02       8110        MVI   C,2
C2DC CD 72 C0    8120        CALL  AHEO
C2DF EB          8130        XCHG
C2E0 4D          8140        MOV   C,L
C2E1 E1          8150        POP   H
C2E2 7E          8160 CONT   MOV   A,M           READ MEMORY
C2E3 B8          8170        CMP   B             COMPARE TO CODE
C2E4 C2 FB C2    8180        JNZ   SKP           SKIP IF NO COMP
C2E7 F1          8190        POP   PSW           FETCH CONTROL
C2E8 FE 53       8200        CPI   'S'
C2EA F5          8210        PUSH  PSW
C2EB CA F5 C2    8220        JZ    OBCP
C2EE 23          8230        INX   H
C2EF 7E          8240        MOV   A,M
C2F0 2B          8250        DCX   H
C2F1 B9          8260        CMP   C
C2F2 C2 FB C2    8270        JNZ   SKP
C2F5 23          8280 OBCP   INX   H
C2F6 7E          8290        MOV   A,M           READ NEXT BYTE
C2F7 2B          8300        DCX   H             DECR ADDRESS
C2F8 CD C8 C1    8310        CALL  ERR           PRINT CODES
C2FB CD 54 C2    8320 SKP    CALL  BMP           CHECK IF DONE
C2FE C2 E2 C2    8330        JNZ   CONT          BACK FOR MORE
C301 F1          8340        POP   PSW
C302 C9          8350        RET
C303             8360 * SHORT ROUTINE TO SAVE CODE
C303 CD 70 C0    8370 TAHEX  CALL  AHEX
C306 CD 70 C0    8380        CALL  AHEX
C309 C9          8390        RET
C30A             8400 * OUTPUT DATA TO A PORT
C30A 0E 02       8410 POUTP  MVI   C,2
C30C CD 72 C0    8420        CALL  AHEO          PORT NO IN E
C30F 0E 02       8430        MVI   C,2
C311 CD 72 C0    8440        CALL  AHEO          PORT TO L DATA E
C314 55          8450        MOV   D,L
C315 21 A0 DF    8460        LXI   H,SPTR-30H
C318 36 C9       8470        MVI   M,0C9H
C31A 2B          8480        DCX   H
C31B 72          8490        MOV   M,D
C31C 2B          8500        DCX   H
C31D 36 D3       8510        MVI   M,0D3H
C31F 7B          8520        MOV   A,E
C320 E9          8530        PCHL

C321             8540 * INPUT DATA FROM A PORT
C321 0E 02       8550 PINPT  MVI   C,2
C323 CD 72 C0    8560        CALL  AHEO          PORT NO TO E
C326 21 A0 DF    8570        LXI   H,SPTR-30H
C329 36 C9       8580        MVI   M,0C9H        RETURN
C32B 2B          8590        DCX   H
C32C 73          8600        MOV   M,E           PORT NO
C32D 2B          8610        DCX   H
C32E 36 DB       8620        MVI   M,0DBH
C330 CD 9E DF    8630        CALL  SPTR-32H
C333 C3 D4 C1    8640        JMP   PT2
C336             8650 * OUTPUT TARBELL SYNC
C336 3E E6       8660 SYNC   MVI   A,0E6H        SYNC BYTE
C338 4F          8670        MOV   C,A
C339 CD 13 C1    8680        CALL  COUT          OUTPUT TO CASS
C33C CD DC C0    8690        CALL  CNTLC
C33F C3 36 C3    8700        JMP   SYNC
C342             8710 * THE VECTOR 1 RELOCATING LOADER
C342 3E 10       8720 RLODR  MVI   A,10H         RESET CASS. INT.
C344 D3 6E       8730        OUT   CASC
C346 CD 03 C3    8740        CALL  TAHEX
C349 E5          8750        PUSH  H             SAVE START ADD.
C34A 06 00       8760        MVI   B,0           ZERO CHECKSUM
C34C CD 69 C1    8770 PASS1  CALL  CIN           READ CASS.
C34F 77          8780        MOV   M,A           WRITE IN MEMORY
C350 BE          8790        CMP   M             CHECK IF THERE
C351 C2 A8 C3    8800        JNZ   MERR          MEMORY ERROR
C354 80          8810        ADD   B             ADD TO CHECKSUM
C355 47          8820        MOV   B,A           PUT IT BACK
C356 CD 54 C2    8830        CALL  BMP           ALL DONE
C359 C2 4C C3    8840        JNZ   PASS1         LOOP IF NOT
C35C CD 69 C1    8850        CALL  CIN           READ LAST BYTE
C35F B8          8860        CMP   B             CHECKSUM AGREE?
C360 C2 A2 C3    8870        JNZ   LDERR
C363 3E 10       8880 PASS2  MVI   A,10H
C365 D3 6E       8890        OUT   CASC          RESET CASS.
C367 E1          8900        POP   H             GET START ADD.
C368 E5          8910        PUSH  H             SAVE IT BACK
C369 06 00       8920        MVI   B,0           ZERO CHECKSUM
C36B CD 69 C1    8930 L2LP   CALL  CIN           READ CASS.
C36E F5          8940        PUSH  PSW
C36F 80          8950        ADD   B             ADD TO CHECKSUM
C370 47          8960        MOV   B,A           PUT IT BACK
C371 F1          8970        POP   PSW
C372 BE          8980        CMP   M
C373 CA 8F C3    8990        JZ    OK
C376 96          9000        SUB   M
C377 FE 17       9010        CPI   17H           IS IT LSB?
C379 C2 8A C3    9020        JNZ   HIBYT         JUMP IF NOT
C37C E3          9030        XTHL
C37D 7D          9040        MOV   A,L           LO BYTE OF START
C37E E3          9050        XTHL
C37F 86          9060        ADD   M
C380 77           9070       MOV   M,A           REPLACE NEW ADD.
C381 D2 8F C3    9080        JNC   OK            NO OVERFLOW
C384 23          9090        INX   H             LOOK AT HI BYTE
```

```
C385 34                  9100          INR   M             BUMP IT
C386 2B                  9110          DCX   H             RESTORE H
C387 C3 8F C3            9120          JMP   OK
C38A E3                  9130 HIBYT    XTHL
C38B 7C                  9140          MOV   A,H           HI BYTE OF START
C38C E3                  9150          XTHL
C38D 86                  9160          ADD   M
C38E 77                  9170          MOV   M,A           REPLACE NEW AD.
C38F CD 54 C2            9180 OK       CALL  BMP           CHECK IF DONE
C392 C2 6B C3            9190          JNZ   L2LP          LOOP IF NOT
C395 CD 69 C1            9200          CALL  CIN           READ LAST BYTE
C398 B8                  9210          CMP   B             CHECKSUM AGREE?
C399 C2 A2 C3            9220          JNZ   LDERR         BAD(LOAD
C39C 21 CC C3            9230 GDLD     LXI   H,M3
C39F C3 AB C3            9240          JMP   ERMSG
C3A2 21 B6 C3            9250 LDERR    LXI   H,M1
C3A5 C3 AB C3            9260          JMP   ERMSG
C3A8 21 BF C3            9270 MERR     LXI   H,M2
C3AB F1                  9280 ERMSG    POP   PSW           CLEAR STACK
C3AC 7E                  9290          MOV   A,M           READ MESSAGE
C3AD CD 98 C0            9300          CALL  PTCN
C3B0 A7                  9310          ANA   A             SET FLAGS
C3B1 23                  9320          INX   H
C3B2 C2 AC C3            9330          JNZ   ERMSG+1
C3B5 C9                  9340          RET
C3B6 42 41 44 20 4C      9350 M1       ASC   'BAD LOAD'
     4F 41 44
C3BE 00                  9360          NOP
C3BF 4D 45 4D 4F 52      9370 M2       ASC   'MEMORY ERROR'
     59 20 45 52 52
     4F 52
C3CB 00                  9380          NOP
C3CC 47 4F 4F 44 20      9390 M3       ASC   'GOOD LOAD'
     4C 4F 41 44
C3D5 00                  9400          NOP
C3D6 CA 95 C0            9410 PATCH    JZ    SPCE-1
C3D9 FE 30               9420          CPI   30H
C3DB DA 0B C0            9430          JC    START
C3DE C3 7D C0            9440          JMP   PTHRT
C3E1                     9450 *
C3E1                     9460 * BOOT LOAD N.STAR OR MICROPOLIS
C3E1                     9470 *
C3E1 3A 00 E9            9480 BOOT     LDA   0E900H
C3E4 FE 31               9490          CPI   31H
C3E6 CA 00 E9            9500          JZ    0E900H
C3E9 3A 00 F4            9502          LDA   0F400H
C3EC FE F3               9504          CPI   0F3H
C3EE CA 00 F4            9506          JZ    0F400H
C3F1 C3 00 D8            9508          JMP   0D800H
C3F4                     9530 *
C3F4                     9540 * WARM START
C3F4                     9550 *
C3F4 21 28 20            9560 WARM     LXI   H,2028H
C3F7 7E                  9570          MOV   A,M
C3F8 FE C3               9580          CPI   0C3H
C3FA C2 E7 04            9590          JNZ   04E7H         MICR. RESTART
C3FD E9                  9600          PCHL                N.STAR WM. ST.
C3FE                     9610 *

SYMBOL TABLE

AHEO  C072   AHE1  C075   AHEX  C070   ALF   C08F   ALPH  C0B0
ASCD  C20F   BINH  C1DC   BINL  C1E0   BMP   C254   BOOT  C3E1
CASC  006E   CASD  006F   CERR  C163   CHKSM C124   CILOP C135
CIN   C169   CINO  C142   CINR  C128   CLOP  C11A   CMDTB C037
CNTLC C0DC   COLOP C0FD   COMPR C2AC   CON1  C246   CON2  C250
COND  0002   CONS  0003   CONT  C2E2   COUT  C113   COUTR C0ED
CRIG  C233   CRLF  C0A6   CYCL  C17C   DISP  C1EE   ENT1  C1F2
ERMSG C3AB   ERR   C1C8   EXEC  C06B   GDLD  C39C   GOON  C25B
HIBYT C38A   INIT  C003   L2LP  C36B   LDERR C3A2   LP2   C1F7
LP3   C204   M1    C3B6   M2    C3BF   M3    C3CC   MERR  C3A8
MLOOP C27B   MONIT C000   MOVEB C271   NCON  C21A   NDLOP C296
NDMT  C293   NEXCH C287   OBCP  C2F5   OK    C38F   PASS1 C34C
PASS2 C363   PATCH C3D6   PAUSE C0CD   PEVE  C1AF   PGLP  C22A
PGM   C223   PINPT C321   PLOOP C0D3   POUTP C30A   PT2   C1D4
PTAD  C1B6   PTCN  C098   PTHRT C07D   PTLOP C099   RDA   0002
RDCN  C0BD   RLODR C342   RLOP  C191   RNDM  C1A4   SKP   C2FB
SPCE  C096   SPTR  DFD0   SRCH  C2C5   START C00B   STPOL 0000
SYNC  C336   TAHEX C303   TBE   0001   TLOP  C182   TMEM  C176
VMLOP C2B4   WARM  C3F4   ZEROM C25D   ZLOOP C269
```

```
C000                    0010 *** VECTOR ONE MONITOR - VERSION 2.1
C000                    0020 * R. S. HARP 1/15/78
C000                    0030 * OPTION EV (FLASHWRITER I/O)
C000                    0040 *
C000                    0050 * SYSTEM EQUATES
C000                    0060 CONS    EQU   0            CONS STATUS PRT
C000                    0070 COND    EQU   1            CONS DATA PORT
C000                    0090 RDA     EQU   40H          RECEIVE FLAG
C000                    0100 STPOL   EQU   0            STATUS POLARITY
C000                    0110 CASD    EQU   6FH          CASS DATA PORT
C000                    0120 CASC    EQU   6EH          CASS STAT PORT
C000                    0130 SPTR    EQU   0DFD0H       STACK POINTER
C000                    0150 *
C000                    0160 ********* COMMAND FORMAT *********
C000                    0170 * A SSSS FFFF ASCII DUMP
C000                    0180 * B JUMP TO BOOTSTRAP LOADER
C000                    0190 * C SSSS FFFF CCCC COMPARE BLOCKS
C000                    0200 * D SSSS FFFF DUMP IN HEX
C000                    0210 * E JUMP TO EXTENSION PROM
C000                    0220 * F SSSS FFFF DD DD TWO BYTE SEARCH
C000                    0230 * G SSSS GO TO AND EXECUTE
C000                    0235 * H JUMP TO 2A00
C000                    0240 * I PP INPUT FROM PORT
C000                    0250 * J JUMP TO DOS
C000                    0260 * K JUMP TO 0000
C000                    0270 * L SSSS FFFF LOAD AND GO
C000                    0280 * M SSSS FFFF DDDD MOVE BLOCK
C000                    0290 * N NON DESTRUCTIVE MEMORY TEST
C000                    0300 * O PP DD OUTPUT TO PORT
C000                    0310 * P LLLL PROGRAM MEMORY
C000                    0320 * Q SSSS FFFF COMPUTE CHECKSUM
C000                    0330 * R SSSS FFFF READ CASSETTE
C000                    0340 * S SSSS FFFF DD SEARCH FOR SINGLE BYTE
C000                    0350 * T SSSS FFFF TEST MEMORY
C000                    0360 * U GENERATES TARBELL SYNC
C000                    0370 * V SSSS FFFF VERIFY TAPE
C000                    0380 * W SSSS FFFF WRITE TAPE
C000                    0390 * X SSSS FFFF DDDD EXCHANGE BLOCK
C000                    0400 * Y KEYBOARD ECHO
C000                    0410 * Z SSSS FFFF DD ZERO OR FILL MEMORY
C000                    0420 ************************************
C000                    0430 *
C000 C3 03 C0           0440 MONIT   JMP   INIT
C003 DB 01              0450 INIT    IN    COND
C005                    0455         DS    6
C00B 31 D0 DF           0490 START   LXI   SP,SPTR
C00E CD A6 C0           0500         CALL  CRLF
C011 3E 2A              0510         MVI   A,'*'
C013 CD 98 C0           0520         CALL  PTCN
C016 CD BD C0           0530         CALL  RDCN              READ KEYBOARD
C019 E6 5F              0540         ANI   5FH               UPPER AND LOWER
C01B F5                 0550         PUSH  PSW               SAVE INPUT
C01C CD 96 C0           0560         CALL  SPCE
C01F F1                 0570         POP   PSW               SAVE INPUT
C020 21 0B C0           0580         LXI   H,START
C023 E5                 0590         PUSH  H
C024 FE 41              0600         CPI   'A'
C026 D8                 0610         RC                      TOO SMALL
C027 FE 5B              0620         CPI   05BH
C029 D0                 0630         RNC                     TOO LARGE
C02A 21 B5 C0           0640         LXI   H,CMDTB+7EH
C02D F5                 0650         PUSH  PSW
C02E 87                 0660         ADD   A
C02F 85                 0670         ADD   L
C030 6F                 0680         MOV   L,A
C031 5E                 0690         MOV   E,M
C032 23                 0700         INX   H
C033 56                 0710         MOV   D,M
C034 EB                 0720         XCHG
C035 F1                 0730         POP   PSW
C036 E9                 0740         PCHL                    AWAY WE GO
C037                    0750 * COMMAND TABLE
C037 EE C1              0760 CMDTB   DW    DISP              A
C039 4D C3              0770         DW    BOOT              B
C03B AC C2              0780         DW    COMPR             C
C03D EE C1              0790         DW    DISP              D
C03F 00 C4              0800         DW    MONIT+0400H       E
C041 C5 C2              0810         DW    SRCH              F
C043 6B C0              0820         DW    EXEC              G
C045 00 2A              0830         DW    02A00H            H
C047 21 C3              0840         DW    PINPT             I
C049 60 C3              0850         DW    WARM              J
C04B 00 00              0860         DW    0                 K
C04D 28 C1              0870         DW    CINR              L
C04F 71 C2              0880         DW    MOVEB             M
C051 93 C2              0890         DW    NDMT              N
C053 0A C3              0900         DW    POUTP             O
C055 23 C2              0910         DW    PGM               P
C057 ED C0              0920         DW    COUTR             Q
C059 28 C1              0930         DW    CINR              R
C05B C5 C2              0940         DW    SRCH              S
C05D 76 C1              0950         DW    TMEM              T
C05F 36 C3              0960         DW    SYNC              U
C061 28 C1              0970         DW    CINR              V
C063 ED C0              0980         DW    COUTR             W
C065 71 C2              0990         DW    MOVEB             X
C067 F2 C3              1000         DW    ECHO              Y
C069 5D C2              1010         DW    ZEROM             Z
C06B                    1020 *
C06B                    1030 *** EXECUTE THE PROGRAM AT THE ADDRESS ***
C06B                    1040 *
C06B CD 70 C0           1050 EXEC    CALL  AHEX              READ ADD FROM KB
C06E EB                 1060         XCHG
C06F E9                 1070         PCHL                    JUMP TO IT
C070                    1080 *
C070                    1090 *** CONVERT UP TO 4 HEX DIGITS TO BIN
C070                    1100 *
C070 0E 04              1110 AHEX    MVI   C,4               COUNT OF 4 DIGITS
C072 21 00 00           1120 AHE0    LXI   H,0               16 BIT ZERO
C075 CD BD C0           1130 AHE1    CALL  RDCN
```

```
C078 FE 20
C07A C3 42 C3
C07D FE 3A
C07F D4 B0 C0
C082 29
C083 29
C084 29
C085 29
C086 D6 30
C088 FE 0A
C08A DA 8F C0
C08D D6 07
C08F 85
C090 6F
C091 0D
C092 C2 75 C0
C095 EB
C096 3E 20
C098 C3 6A C3
C09B EE 00
C09D E6 00
C09F CA 0B C0
C0A2 F1
C0A3 D3 01
C0A5 C9
C0A6 3E 0D
C0A8 CD 98 C0
C0AB 3E 0A
C0AD C3 98 C0
C0B0 FE 41
C0B2 DA 0B C0
C0B5 E6 5F
C0B7 FE 47
C0B9 D2 0B C0
C0BC C9
C0BD
C0BD
C0BD
C0BD CD DC C0
C0C0 CA BD C0
C0C3 FE 60
C0C5 DA 98 C0
C0C8 E6 5F
C0CA C3 98 C0
C0CD CD DC C0
C0D0 FE 20
C0D2 C0
C0D3 CD DC C0
C0D6 FE 20
C0D8 C2 D3 C0
C0DB C9
C0DC DB 00
C0DE EE 00
C0E0 E6 40
C0E2 C8
C0E3 DB 01
```

```
1140          CPI   20H
1150          JMP   PATCH
1160 PTHRT    CPI   ':'
1170          CNC   ALPH
1180          DAD   H              SHIFT 4 LEFT
1190          DAD   H
1200          DAD   H
1210          DAD   H
1220          SUI   48             ASCII BIAS
1230          CPI   10             DIGIT 0-10
1240          JC    ALF
1250          SUI   7              ALPHA BIAS
1260 ALF      ADD   L
1270          MOV   L,A
1280          DCR   C              4 DIGITS?
1290          JNZ   AHE1           KEEP READING
1300          XCHG
1310 SPCE     MVI   A,20H          PRINT SPACE
1320 PTCN     JMP   VIDEO
1340          XRI   STPOL
1350          ANI   0
1360          JZ    START
1370          POP   PSW            THEN RECOVER A
1380          OUT   COND           AND PRINT IT
1390          RET                  RETURN FROM PTCN
1400 CRLF     MVI   A,0DH          PRINT CR
1410          CALL  PTCN
1420          MVI   A,0AH
1430          JMP   PTCN
1440 ALPH     CPI   'A'
1450          JC    START
1460          ANI   5FH
1470          CPI   'G'
1480          JNC   START
1490          RET
1500 *
1510 *** READ FROM CONSOLE TO REG A ***
1520 *
1530 RDCN     CALL  CNTLC          READ KEYBOARD
1540          JZ    RDCN
1550          CPI   60H
1560          JC    PTCN
1570          ANI   5FH
1580          JMP   PTCN
1590 PAUSE    CALL  CNTLC
1600          CPI   20H
1610          RNZ
1620 PLOOP    CALL  CNTLC
1630          CPI   20H
1640          JNZ   PLOOP
1650          RET
1660 CNTLC    IN    CONS
1670          XRI   STPOL
1680          ANI   RDA
1690          RZ
1700          IN    COND
```

```
C0E5 E6 7F
C0E7 FE 18
C0E9 CA 0B C0
C0EC C9
C0ED
C0ED
C0ED F5
C0EE CD 03 C3
C0F1 06 00
C0F3 F1
C0F4 4F
C0F5 CD 13 C1
C0F8 3E E6
C0FA CD 13 C1
C0FD 7E
C0FE CD 13 C1
C101 80
C102 47
C103 CD 54 C2
C106 C2 FD C0
C109 78
C10A CD 13 C1
C10D CD D4 C1
C110 C3 0B C0
C113 F5
C114 79
C115 FE 51
C117 CA 24 C1
C11A CD CD C0
C11D DB 6E
C11F E6 20
C121 C2 1A C1
C124 F1
C125 D3 6F
C127 C9
C128
C128
C128
C128 F5
C129 3E 10
C12B D3 6E
C12D CD 03 C3
C130 F1
C131 E5
C132 F5
C133 06 00
C135 CD 69 C1
C138 4F
C139 F1
C13A F5
C13B FE 56
C13D 79
C13E CA 42 C1
C141 77
C142 80
C143 47
```

```
1710          ANI   7FH
1720          CPI   18H            CNTL X
1730          JZ    START
1740          RET
1750 *** CASSETTE INTERFACE OUTPUT ROUTINE ***
1760 *
1770 COUTR    PUSH  PSW            SAVE CONTROL
1780          CALL  TAHEX
1790          MVI   B,0            START CHECKSUM = 0
1800          POP   PSW
1810          MOV   C,A
1820          CALL  COUT           START BYTE OUT
1830          MVI   A,0E6H         SEND SYNC BYTE
1840          CALL  COUT           TO CASSETTE
1850 COLOP    MOV   A,M            GET DATA FROM MEM
1860          CALL  COUT           SEND TO CASSETTE
1870          ADD   B              ADD TO CHECKSUM
1880          MOV   B,A
1890          CALL  BMP
1900          JNZ   COLOP          REPEAT LOOP
1910          MOV   A,B            GET CHECKSUM
1920          CALL  COUT           OUTPUT IT
1930          CALL  PT2            PRINT CHECKSUM
1940          JMP   START          GET ANOTH COMMND
1950 COUT     PUSH  PSW            SAVE A AND FLAGS
1960          MOV   A,C
1970          CPI   'Q'
1980          JZ    CHKSM
1990 CLOP     CALL  PAUSE
2000          IN    CASC           READ CASS STATUS
2010          ANI   20H            LOOK AT BIT 5
2020          JNZ   CLOP           TRY AGAIN?
2030 CHKSM    POP   PSW
2040          OUT   CASD           SEND DATA TO CASS
2050          RET   RETURN         FROM COUT
2060 *
2070 *** CASSETTE INPUT ROUTINE ***
2080 *
2090 CINR     PUSH  PSW            SAVE CONTROL CHAR
2100          MVI   A,10H          USE BIT 4 IN REG A
2110          OUT   CASC           TO RESET CASS INT
2120          CALL  TAHEX          READ ADDRESSES
2130          POP   PSW            GET CONTROL CHAR
2140          PUSH  H              SAVE START ADD
2150          PUSH  PSW            UNDER CONTROL CHAR
2160          MVI   B,0            SET CHECKSUM = 0
2170 CILOP    CALL  CIN            READ FM CONS
2180          MOV   C,A            SAVE IT IN REG C
2190          POP   PSW            GET CONTROL CHAR
2200          PUSH  PSW            SAVE IT BACK
2210          CPI   'V'            IS IT A V?
2220          MOV   A,C            GET BACK DATA BYTE
2230          JZ    CINO           IF C, DON'T STORE
2240          MOV   M,A            IF NOT , STORE
2250 CINO     ADD   B              ADD TO CHECKSUM
2260          MOV   B,A
```

```
C144 CD 54 C2          2270         CALL   BMP
C147 C2 35 C1          2280         JNZ    CILOP          READ MORE
C14A CD 69 C1          2290         CALL   CIN            READ LAST BYTE
C14D F5                2300         PUSH   PSW
C14E CD D4 C1          2310         CALL   PT2            PRINT CHECKSUM
C151 CD 96 C0          2320         CALL   SPCE           SPACE OVER
C154 F1                2330         POP    PSW
C155 B8                2340         CMP    B              COMP TO CHKSUM
C156 3E 45             2350         MVI    A,'E'          PRINT E FOR ERROR
C158 C2 63 C1          2360         JNZ    CERR           PRINT NOW IF ERROR
C15B F1                2370         POP    PSW            RECOVER CTL CHAR
C15C FE 4C             2380         CPI    'L'            IF NOT L
C15E C2 63 C1          2390         JNZ    CERR           DON'T EXECUTE
C161 E1                2400         POP    H              OTHERWISE, EXECJTE
C162 E9                2410         PCHL   AT             STARTING ADDRESS
C163 CD 98 C0          2420 CERR    CALL   PTCN           PRINT V,E, OR R
C166 C3 0B C0          2430         JMP    START
C169 CD CD C0          2440 CIN     CALL   PAUSE
C16C DB 6E             2450         IN     CASC           READ STATUS
C16E E6 10             2460         ANI    10H            LOOK AT BIT 4
C170 C2 69 C1          2470         JNZ    CIN            WAIT UNTIL LOW
C173 DB 6F             2480         IN     CASD           READ DATA FM CASS
C175 C9                2490         RET    RETURN         FROM CIN
C176                   2500 *
C176                   2510 *** MEMORY TEST ROUTINE ***
C176                   2520 *
C176 CD 03 C3          2530 TMEM    CALL   TAHEX          READ ADDRESSES
C179 01 5A 5A          2540         LXI    B,5A5AH        INI B,C
C17C CD A4 C1          2550 CYCL    CALL   RNDM
C17F C5                2560         PUSH   B              KEEP ALL REGS
C180 E5                2570         PUSH   H
C181 D5                2580         PUSH   D
C182 CD A4 C1          2590 TLOP    CALL   RNDM
C185 70                2600         MOV    M,B            WRITE IN MEM
C186 CD 54 C2          2610         CALL   BMP
C189 C2 82 C1          2620         JNZ    TLOP           REPEAT LOOP
C18C D1                2630         POP    D
C18D E1                2640         POP    H              RESTORE ORIG
C18E C1                2650         POP    B              VALUES OF
C18F E5                2660         PUSH   H
C190 D5                2670         PUSH   D
C191 CD A4 C1          2680 RLOP    CALL   RNDM           GEN NEW SEQ
C194 7E                2690         MOV    A,M            READ MEM
C195 B8                2700         CMP    B              COMP MEM
C196 C4 C8 C1          2710         CNZ    ERR            CALL ERROR ROUT
C199 CD 54 C2          2720         CALL   BMP
C19C C2 91 C1          2730         JNZ    RLOP
C19F D1                2740         POP    D
C1A0 E1                2750         POP    H
C1A1 C3 7C C1          2760         JMP    CYCL
C1A4                   2770 *** THIS ROUTINE GENERATES RANDOM NOS ***
C1A4 CD CD C0          2780 RNDM    CALL   PAUSE
C1A7 78                2790         MOV    A,B            LOOK AT B
C1A8 E6 B4             2800         ANI    0B4H           MASK BITS
C1AA A7                2810         ANA    A              CLEAR CY
C1AB EA AF C1          2820         JPE    PEVE           JUMP IF EVEN
C1AE 37                2830         STC
C1AF 79                2840 PEVE    MOV    A,C            LOOK AT C
C1B0 17                2850         RAL                   ROTATE CY IN
C1B1 4F                2860         MOV    C,A            RESTORE C
C1B2 78                2870         MOV    A,B            LOOK AT B
C1B3 17                2880         RAL                   ROTATE CY IN
C1B4 47                2890         MOV    B,A            RESTORE B
C1B5 C9                2900         RET                   RETURN W NEW B,C
C1B6                   2910 *
C1B6                   2920 *** ERROR PRINT OUT ROUTINE
C1B6                   2930 *
C1B6 CD A6 C0          2940 PTAD    CALL   CRLF           PRINT CR,LF
C1B9 CD CD C0          2950         CALL   PAUSE
C1BC 7C                2960         MOV    A,H            PRINT
C1BD CD D4 C1          2970         CALL   PT2            ASCII
C1C0 7D                2980         MOV    A,L            CODES
C1C1 CD D4 C1          2990         CALL   PT2            FOR
C1C4 CD 96 C0          3000         CALL   SPCE           ADDRESS
C1C7 C9                3010         RET
C1C8 F5                3020 ERR     PUSH   PSW            SAVE ACC
C1C9 CD B6 C1          3030         CALL   PTAD           PRINT ADD.
C1CC 78                3040         MOV    A,B            DATA
C1CD CD D4 C1          3050         CALL   PT2            WRITTEN
C1D0 CD 96 C0          3060         CALL   SPCE
C1D3 F1                3070         POP    PSW            DATA READ
C1D4 F5                3080 PT2     PUSH   PSW
C1D5 CD DC C1          3090         CALL   BINH
C1D8 F1                3100         POP    PSW
C1D9 C3 E0 C1          3110         JMP    BINL
C1DC 1F                3120 BINH    RAR
C1DD 1F                3130         RAR
C1DE 1F                3140         RAR
C1DF 1F                3150         RAR
C1E0 E6 0F             3160 BINL    ANI    0FH            LOW 4 BITS
C1E2 C6 30             3170         ADI    48             ASCII BIAS
C1E4 FE 3A             3180         CPI    58             DIGIT 0-9
C1E6 DA 98 C0          3190         JC     PTCN
C1E9 C6 07             3200         ADI    7              DIGIT A-F
C1EB C3 98 C0          3210         JMP    PTCN
C1EE                   3220 *
C1EE                   3230 *** DISPLAY MEMORY CONTENTS ***
C1EE                   3240 *
C1EE 47                3250 DISP    MOV    B,A            SAVE CONTROL
C1EF CD 03 C3          3260         CALL   TAHEX          READ ADDRESSES
C1F2 0E 10             3270 ENT1    MVI    C,16           LOC/LINE
C1F4 CD B6 C1          3280         CALL   PTAD
C1F7 78                3290 LP2     MOV    A,B
C1F8 FE 41             3300         CPI    'A'            IS IT "A"?
C1FA 7E                3310         MOV    A,M
C1FB CA 0F C2          3320         JZ     ASCD           DUMP ASCII
C1FE CD D4 C1          3330         CALL   PT2            PRINT OUT
C201 CD 96 C0          3340         CALL   SPCE
C204 CD 54 C2          3350 LP3     CALL   BMP
C207 C8                3360         RZ
C208 0D                3370         DCR    C
C209 CA F2 C1          3380         JZ     ENT1           END OF LINE
```

```
C20C C3 F7 C1   3390         JMP   LP2           CONTINUE LOOP
C20F E6 60      3400 ASCD    ANI   60H           MASK FOR CONTROL
C211 C2 1A C2   3410         JNZ   NCON
C214 CD 96 C0   3420         CALL  SPCE
C217 C3 04 C2   3430         JMP   LP3
C21A 7E         3440 NCON    MOV   A,M
C21B E6 7F      3450         ANI   7FH           MASK FOR ASCII
C21D CD 98 C0   3460         CALL  PTCN
C220 C3 04 C2   3470         JMP   LP3
C223            3480 *
C223            3490 *** PROGRAM MEMORY *****
C223            3500 *
C223 CD 70 C0   3510 PGM     CALL  AHEX          READ ADD.
C226 EB         3520         XCHG
C227 CD A6 C0   3530         CALL  CRLF
C22A 7E         3540 PGLP    MOV   A,M           READ MEMORY
C22B CD D4 C1   3550         CALL  PT2           PRINT 2 DIG.
C22E 3E 2D      3560         MVI   A,'-'         LOAD DASH
C230 CD 98 C0   3570         CALL  PTCN          PRINT DASH
C233 CD BD C0   3580 CRIG    CALL  RDCN
C236 FE 20      3590         CPI   20H           SPACE
C238 CA 50 C2   3600         JZ    CON2          SKIP IF SPACE
C23B FE 0D      3610         CPI   0DH
C23D C2 46 C2   3620         JNZ   CON1          SKIP IF CR
C240 CD A6 C0   3630         CALL  CRLF          PRINT CR,LF
C243 C3 33 C2   3640         JMP   CRIG          BACK FO MO
C246 EB         3650 CON1    XCHG                H,L>D,E
C247 21 00 00   3660         LXI   H,0           GET 16 BIT ZERO
C24A 0E 02      3670         MVI   C,2           COUNT 2 DIG.
C24C CD 78 C0   3680         CALL  AHE1+3        CONV TO HEX
C24F 73         3690         MOV   M,E           WRITE IN MEM
C250 23         3700 CON2    INX   H             INC POINTER
C251 C3 2A C2   3710         JMP   PGLP          KEEP GOING
C254            3720 * COMPARE ADDRESSES AND INCREMENT H
C254 7B         3730 BMP     MOV   A,E
C255 95         3740         SUB   L
C256 C2 5B C2   3750         JNZ   GOON
C259 7A         3760         MOV   A,D
C25A 9C         3770         SBB   H
C25B 23         3780 GOON    INX   H
C25C C9         3790         RET
C25D            3800 * ZERO OR FILL MEMORY WITH A CONSTANT
C25D CD 03 C3   3810 ZEROM   CALL  TAHEX         READ ADDRESSES
C260 E5         3820         PUSH  H             SAVE H
C261 0E 02      3830         MVI   C,2           2 DIGITS
C263 CD 72 C0   3840         CALL  AHE0          READ 2 DIGITS
C266 EB         3850         XCHG
C267 E3         3860         XTHL                RESTORE H,L
C268 C1         3870         POP   B
C269 71         3880 ZLOOP   MOV   M,C           WRITE INTO MEM
C26A CD 54 C2   3890         CALL  BMP           COMP ADD, INCREMENT
C26D C8         3900         RZ                  RETURN IF DONE
C26E C3 69 C2   3910         JMP   ZLOOP         CONTINUE TIL DONE
C271            3920 * MOVE A BLOCK OF MEMORY
C271 47         3930 MOVEB   MOV   B,A
C272 CD 03 C3   3940         CALL  TAHEX         READ ADDRESSES
C275 E5         3950         PUSH  H
C276 CD 70 C0   3960         CALL  AHEX
C279 EB         3970         XCHG
C27A E3         3980         XTHL                BACK TO NORMAL
C27B 4E         3990 MLOOP   MOV   C,M
C27C E3         4000         XTHL
C27D 78         4010         MOV   A,B
C27E FE 4D      4020         CPI   'M'
C280 CA 87 C2   4030         JZ    NEXCH
C283 7E         4040         MOV   A,M
C284 E3         4050         XTHL
C285 77         4060         MOV   M,A
C286 E3         4070         XTHL
C287 71         4080 NEXCH   MOV   M,C
C288 23         4090         INX   H
C289 E3         4100         XTHL
C28A CD 54 C2   4110         CALL  BMP
C28D CA 0B C0   4120         JZ    START
C290 C3 7B C2   4130         JMP   MLOOP
C293            4140 * NON DESTRUCTIVE MEMORY TEST
C293 21 00 00   4150 NDMT    LXI   H,0           START AT ZERO
C296 4E         4160 NDLOP   MOV   C,M
C297 06 FF      4170         MVI   B,0FFH
C299 70         4180         MOV   M,B
C29A 7E         4190         MOV   A,M
C29B B8         4200         CMP   B
C29C C2 C8 C1   4210         JNZ   ERR           PRINT ERROR
C29F 06 00      4220         MVI   B,0
C2A1 70         4230         MOV   M,B
C2A2 7E         4240         MOV   A,M
C2A3 B8         4250         CMP   B
C2A4 C2 C8 C1   4260         JNZ   ERR
C2A7 71         4270         MOV   M,C
C2A8 23         4280         INX   H
C2A9 C3 96 C2   4290         JMP   NDLOP
C2AC            4300 * COMPARE TWO BLOCKS OF MEMORY
C2AC CD 03 C3   4310 COMPR   CALL  TAHEX
C2AF E5         4320         PUSH  H
C2B0 CD 70 C0   4330         CALL  AHEX
C2B3 EB         4340         XCHG
C2B4 7E         4350 VMLOP   MOV   A,M
C2B5 23         4360         INX   H
C2B6 E3         4370         XTHL
C2B7 BE         4380         CMP   M
C2B8 46         4390         MOV   B,M
C2B9 C4 C8 C1   4400         CNZ   ERR
C2BC CD 54 C2   4410         CALL  BMP
C2BF E3         4420         XTHL
C2C0 C2 B4 C2   4430         JNZ   VMLOP
C2C3 F1         4440         POP   PSW
C2C4 C9         4450         RET
C2C5            4460 * SEARCH FOR SPECIFIC CODES
C2C5 F5         4470 SRCH    PUSH  PSW
C2C6 CD 03 C3   4480         CALL  TAHEX
C2C9 E5         4490         PUSH  H             SAVE H
C2CA 0E 02      4500         MVI   C,2           COUNT OF 2
```

```
C2CC CD 72 C0          4510          CALL  AHEO          READ 2 DIGITS
C2CF EB                4520          XCHG                H=CODE,D=F
C2D0 45                4530          MOV   B,L           PUT CODE IN B
C2D1 E1                4540          POP   H             RESTORE H
C2D2 F1                4550          POP   PSW
C2D3 FE 53             4560          CPI   'S'
C2D5 F5                4570          PUSH  PSW
C2D6 CA E2 C2          4580          JZ    CONT
C2D9 E5                4590          PUSH  H
C2DA 0E 02             4600          MVI   C,2
C2DC CD 72 C0          4610          CALL  AHEO
C2DF EB                4620          XCHG
C2E0 4D                4630          MOV   C,L
C2E1 E1                4640          POP   H
C2E2 7E                4650 CONT     MOV   A,M           READ MEMORY
C2E3 B8                4660          CMP   B             COMPARE TO CODE
C2E4 C2 FB C2          4670          JNZ   SKP           SKIP IF NO COMP
C2E7 F1                4680          POP   PSW           FETCH CONTROL
C2E8 FE 53             4690          CPI   'S'
C2EA F5                4700          PUSH  PSW
C2EB CA F5 C2          4710          JZ    OBCP
C2EE 23                4720          INX   H
C2EF 7E                4730          MOV   A,M
C2F0 2B                4740          DCX   H
C2F1 B9                4750          CMP   C
C2F2 C2 FB C2          4760          JNZ   SKP
C2F5 23                4770 OBCP     INX   H
C2F6 7E                4780          MOV   A,M           READ NEXT BYTE
C2F7 2B                4790          DCX   H             DECR ADDRESS
C2F8 CD C8 C1          4800          CALL  ERR           PRINT CODES
C2FB CD 54 C2          4810 SKP      CALL  BMP           CHECK IF DONE
C2FE C2 E2 C2          4820          JNZ   CONT          BACK FOR MORE
C301 F1                4830          POP   PSW
C302 C9                4840          RET
C303                   4850 * SHORT ROUTINE TO SAVE CODE
C303 CD 70 C0          4860 TAHEX    CALL  AHEX
C306 CD 70 C0          4870          CALL  AHEX
C309 C9                4880          RET
C30A                   4890 * OUTPUT DATA TO A PORT
C30A 0E 02             4900 POUTP    MVI   C,2
C30C CD 72 C0          4910          CALL  AHEO          PORT NO IN E
C30F 0E 02             4920          MVI   C,2
C311 CD 72 C0          4930          CALL  AHEO          PORT TO L DATA E
C314 55                4940          MOV   D,L
C315 21 A0 DF          4950          LXI   H,SPTR-30H
C318 36 C9             4960          MVI   M,0C9H
C31A 2B                4970          DCX   H
C31B 72                4980          MOV   M,D
C31C 2B                4990          DCX   H
C31D 36 D3             5000          MVI   M,0D3H
C31F 7B                5010          MOV   A,E
C320 E9                5020          PCHL
C321                   5030 * INPUT DATA FROM A PORT
C321 0E 02             5040 PINPT    MVI   C,2
C323 CD 72 C0          5050          CALL  AHEO          PORT NO TO E
C326 21 A0 DF          5060          LXI   H,SPTR-30H
C329 36 C9             5070          MVI   M,0C9H        RETURN
C32B 2B                5080          DCX   H
C32C 73                5090          MOV   M,E           PORT NO
C32D 2B                5100          DCX   H
C32E 36 DB             5110          MVI   M,0DBH
C330 CD 9E DF          5120          CALL  SPTR-32H
C333 C3 D4 C1          5130          JMP   PT2
C336                   5140 * OUTPUT TARBELL SYNC
C336 3E E6             5150 SYNC     MVI   A,0E6H        SYNC BYTE
C338 4F                5160          MOV   C,A
C339 CD 13 C1          5170          CALL  COUT          OUTPUT TO CASS
C33C CD DC C0          5180          CALL  CNTLC
C33F C3 36 C3          5190          JMP   SYNC
C342 CA 95 C0          5900 PATCH    JZ    SPCE-1
C345 FE 30             5910          CPI   30H
C347 DA 0B C0          5920          JC    START
C34A C3 7D C0          5930          JMP   PTHRT
C34D                   5940 *
C34D                   5950 * BOOT LOAD N.STAR OR MICROPOLIS
C34D                   5960 *
C34D 3A 00 E9          5970 BOOT     LDA   0E900H
C350 FE 31             5980          CPI   31H
C352 CA 00 E9          5990          JZ    0E900H        N.S.BOOT
C355 3A 00 F4          6000          LDA   0F400H
C358 FE F3             6010          CPI   0F3H
C35A CA 00 F4          6012          JZ    0F400H        MICROP. BOOT
C35D C3 00 D8          6016          JMP   0D800H
C360                   6020 *
C360                   6030 * WARM START
C360                   6040 *
C360 21 28 20          6050 WARM     LXI   H,2028H
C363 7E                6060          MOV   A,M
C364 FE C3             6070          CPI   0C3H
C366 C2 E7 04          6080          JNZ   04E7H         MICR. RESTART
C369 E9                6090          PCHL                N.STAR WM. ST.
C36A                   6100 *
C36A                   6110 * SHORT FLASHWRITER VIDEO DRIVER *
C36A                   6120 * R.S.HARP 1/22/78
C36A                   6140 *
C36A                   6150 PAGE     EQU   0D000H        SCREEN ADDRESS
C36A                   6160 PSTAT    EQU   0D0H
C36A                   6170 CURS     EQU   SPTR+0BH
C36A                   6180 IFL      EQU   CURS+2
C36A                   6190 * POP PSW AND ENTER HERE **********
C36A F5                6200 VIDEO    PUSH  PSW
C36B C5                6210          PUSH  B
C36C D5                6220          PUSH  D
C36D E5                6230          PUSH  H
C36E E6 7F             6235          ANI   7FH
C370 47                6240          MOV   B,A
C371 21 DD DF          6250          LXI   H,IFL
C374 3E 20             6260          MVI   A,20H
C376 BE                6270          CMP   M
C377 CA 7E C3          6272          JZ    DISPL
C37A 77                6274          MOV   M,A
C37B C3 CD C3          6276          JMP   CLRSC
```

```
C37E                    6280 * DISPLAY A CHARACTER ON THE SCREEN *
C37E 2A DB DF           6290 DISPL  LHLD  CURS
C381 36 20              6300        MVI   M,20H
C383 78                 6310        MOV   A,B
C384 FE 04              6340        CPI   4               (D) CLEAR SCREEN
C386 CA CD C3           6350        JZ    CLRSC
C389 FE 0A              6360        CPI   0AH             (J) LINE FEED
C38B CA E7 C3           6370        JZ    LNFD
C38E FE 5F              6380        CPI   5FH             UNDRLN BK SPCE
C390 CA EE C3           6390        JZ    BKSPC
C393 FE 08              6395        CPI   08              BACKSPACE
C395 CA EE C3           6397        JZ    BKSPC
C398 FE 20              6400        CPI   20H             CONTROL CHAR
C39A DA DC C3           6410        JC    DELAY
C39D E6 7F              6420        ANI   07FH
C39F 77                 6430        MOV   M,A
C3A0 23                 6440        INX   H
C3A1                    6450 * MAKE SURE CURSOR IS ON THE SCREEN *
C3A1 7C                 6460 ONSCR  MOV   A,H
C3A2 FE D4              6470        CPI   PSTAT+4
C3A4 C2 C3 C3           6480        JNZ   RETRN
C3A7 21 00 D0           6490        LXI   H,PAGE
C3AA 11 40 D0           6500        LXI   D,PAGE+64
C3AD                    6510 * IF NOT,SCROLL UP ONE LINE *
C3AD 1A                 6520 SCROL  LDAX  D
C3AE 77                 6530        MOV   M,A
C3AF 13                 6540        INX   D
C3B0 23                 6550        INX   H
C3B1 7A                 6560        MOV   A,D
C3B2 FE D4              6570        CPI   PSTAT+4
C3B4 C2 AD C3           6580        JNZ   SCROL
C3B7                    6590 * CLEAR THE LAST LINE *
C3B7 36 20              6600 CLRLN  MVI   M,020H
C3B9 23                 6610        INX   H
C3BA 7C                 6620        MOV   A,H
C3BB FE D4              6630        CPI   PSTAT+4
C3BD C2 B7 C3           6640        JNZ   CLRLN
C3C0 21 C0 D3           6650        LXI   H,PAGE+960
C3C3                    6660 * RESTORE THE CURSOR AND RETURN *
C3C3 36 A0              6670 RETRN  MVI   M,0A0H
C3C5 22 DB DF           6680        SHLD  CURS
C3C8 E1                 6690        POP   H
C3C9 D1                 6700        POP   D
C3CA C1                 6710        POP   B
C3CB F1                 6720        POP   PSW
C3CC C9                 6730        RET
C3CD                    6740 * CLEAR THE SCREEN *
C3CD 21 00 D0           6750 CLRSC  LXI   H,PAGE
C3D0 36 20              6760 WRSPC  MVI   M,20H           SPACE
C3D2 23                 6770        INX   H
C3D3 7C                 6780        MOV   A,H
C3D4 FE D8              6790        CPI   PSTAT+8
C3D6 C2 D0 C3           6800        JNZ   WRSPC
C3D9 C3 C0 C3           6810        JMP   RETRN-3
C3DC                    6820 * FIXED DELAY ROUTINE *
C3DC 16 05              6830 DELAY  MVI   D,5
C3DE 7A                 6840 DLOOP  MOV   A,D
C3DF A7                 6850        ANA   A
C3E0 1B                 6860        DCX   D
C3E1 C2 DE C3           6870        JNZ   DLOOP
C3E4 C3 C3 C3           6880        JMP   RETRN
C3E7                    6890 * LINE FEED *
C3E7 11 40 00           6900 LNFD   LXI   D,64
C3EA 19                 6910        DAD   D
C3EB C3 A1 C3           6920        JMP   ONSCR
C3EE                    6930 * BACK SPACE *
C3EE 2B                 6940 BKSPC  DCX   H
C3EF C3 C3 C3           6950        JMP   RETRN
C3F2                    6960 * KEYBOARD ECHO ROUTINE
C3F2 CD DC C0           6970 ECHO   CALL  0C0DCH          CONTROL C
C3F5 C4 6A C3           6980        CNZ   VIDEO
C3F8 C3 F2 C3           6990        JMP   ECHO            KEEP LOOPING
C3FB                    7000 *
```

SYMBOL TABLE

AHE0	C072	AHE1	C075	AHEX	C070	ALF	C08F	ALPH	C0B0
ASCD	C20F	BINH	C1DC	BINL	C1E0	BKSPC	C3EE	BMP	C254
BOOT	C34D	CASC	006E	CASD	006F	CERR	C163	CHKSM	C124
CILOP	C135	CIN	C169	CINO	C142	CINR	C128	CLOP	C11A
CLRLN	C3B7	CLRSC	C3CD	CMDTB	C037	CNTLC	C0DC	COLOP	C0FD
COMPR	C2AC	CON1	C246	CON2	C250	COND	0001	CONS	0000
CONT	C2E2	COUT	C113	COUTR	C0ED	CRIG	C233	CRLF	C0A6
CURS	DFDB	CYCL	C17C	DELAY	C3DC	DISP	C1EE	DISPL	C37E
DLOOP	C3DE	ECHO	C3F2	ENT1	C1F2	ERR	C1C8	EXEC	C06B
GOON	C25B	IFL	DFDD	INIT	C003	LNFD	C3E7	LP2	C1F7
LP3	C204	MLOOP	C27B	MONIT	C000	MOVEB	C271	NCON	C21A
NDLOP	C296	NDMT	C293	NEXCH	C287	OBCP	C2F5	ONSCR	C3A1
PAGE	D000	PATCH	C342	PAUSE	C0CD	PEVE	C1AF	PGLP	C22A
PGM	C223	PINPT	C321	PLOOP	C0D3	POUTP	C30A	PSTAT	00D0
PT2	C1D4	PTAD	C1B6	PTCN	C098	PTHRT	C07D	RDA	0040
RDCN	C0BD	RETRN	C3C3	RLOP	C191	RNDM	C1A4	SCROL	C3AD
SKP	C2FB	SPCE	C096	SPTR	DFD0	SRCH	C2C5	START	C00B
STPOL	0000	SYNC	C336	TAHEX	C303	TLOP	C182	TMEM	C176
VIDEO	C36A	VMLOP	C2B4	WARM	C360	WRSPC	C3D0	ZEROM	C25D
ZLOOP	C269								

VECTOR GRAPHIC EXTENDED MONITOR

The VECTOR GRAPHIC EXTENDED MONITOR available on 2708 or 1702 type proms provides 25 useful commands. When used with VECTOR GRAPHIC Prom/Ram boards system start up is greatly simplified - just turn the power on! The EXTENDED MONITOR is programmed on one 2708 or four 1702A (five with video driver program) and is available for the most popular I/O configurations:

OPTION A	MITS SIO REV 1/3P+S W/INVERTED STATUS
OPTION B	MITS 2 SIO
OPTION C	VECTOR GRAPHIC BITSTREAMER SERIAL IN/OUT IMSAI SIO2
OPTION D	VERSION 1.0 POLY VTI (OBSOLETE)
OPTION E	3P+S WITHOUT INVERTED STATUS BIT
OPTION AV	VDM-1 OUTPUT, INPUT SAME AS OPTION A
OPTION EV	VECTOR GRAPHIC FLASHWRITER ; 3P+S INPUT VDM-1 OUT; VECTOR GRAPHIC BITSTREAMER PARALLEL INPUT, VDM-1 OUT
OPTION P	POLY VTI VERSION 1.2
OPTION EP	3P+S INPUT, POLY VTI 1.2 OUT; BITSTREAMER PARALLEL INPUT, VTI 1.2 OUT
OPTION CV	BITSTREAMER SERIAL IN, FLASHWRITER OR VDM-1 OUTPUT.

EXTENDED MONITOR COMMANDS

A	ASCII DUMP
B	JUMP TO BOOTSTRAP LOADER (I.E. E900H)
C	COMPARE BLOCKS OF MEMORY
D	JUMP MEMORY IN HEX
E	JUMP TO EXTENSION PROM (I.E. C400)
F	FIND TWO BYTES
G	GO TO AND EXECUTE
I	INPUT FROM PORT
J	JUMP TO LOADED DOS (I.E. 2028H)
L	LOAD TARBELL CASSETTE AND GO
M	MOVE MEMORY BLOCK
N	NON DESTRUCTIVE MEMORY TEST
O	OUTPUT TO PORT
P	PROGRAM MEMORY
Q	COMPUTE CHECKSUM
R	READ CASSETTE
S	SEARCH FOR SINGLE BYTE
T	TEST MEMORY
U	TARBELL SYNC
V	VERIFY TAPE
W	WRITE CASSETTE TAPE
X	EXCHANGE MEMORY BLOCKS
Y	RELOCATING LOADER (1702 PROMS)
Y	KEYBOARD ECHO (2708 PROMS)
Z	ZERO OR FILL MEMORY

48K DYNAMIC MEMORY BOARD

Revision 3

USERS MANUAL

Revision B

July 19, 1979

Copyright 1979 Vector Graphic Inc.

Copyright 1979 by Vector Graphic Inc.
All rights reserved.

Disclaimer

Vector Graphic makes no representations or warranties with respect to the contents of this manual itself, whether or not the product it describes is covered by a warranty or repair agreement. Further, Vector Graphic reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Vector Graphic to notify any person of such revision or changes, except when an agreement to the contrary exists.

Revisions

The date and revision of each page herein appears at the bottom of each page. The revision letter such as A or B changes if the MANUAL has been improved but the PRODUCT itself has not been significantly modified. The date and revision on the Title Page corresponds to that of the page most recently revised. When the product itself is modified significantly, the product will get a new revision number, as shown on the manual's title page, and the manual will revert to revision A, as if it were treating a brand new product. EACH MANUAL SHOULD ONLY BE USED WITH THE PRODUCT IDENTIFIED ON THE TITLE PAGE.

Rev. 3-B 7/19/79

TABLE OF CONTENTS

Section		Page
	Table of Contents	1
I.	Introduction	2
	1.1 Description of the Board	2
	1.2 Description of this Manual	2
II.	Theory of Operation	3
	2.1 The 48K Board Circuitry	3
	2.2 The Refresh Feature of the Z-80 Board	4
III.	User's Guide	6
	3.1 Modifications to the 48K Board and the Z-80 Board	6
	3.1.1 Refresh Signal	6
	3.1.2 Reset Circuitry	7
	3.1.2.1 V.G. Rev. 1 Z-80 Board	7
	3.1.2.2 V.G. Rev. 2 Z-80 Board	7
	3.1.2.3 Northstar Z-80 Board	8
	3.1.2.4 Cromemco ZPU Board	8
	3.2 The READY Line	8
	3.3 Using DMA	9
	3.4 Address of the Board	9
	3.5 Memory Test	9
IV.	Specifications	10
V.	Memory Chip Locations	11
	Board Silkscreen	12
	Schematic	13

I. INTRODUCTION

1.1 Description of the Board

The Vector Graphic 48K Dynamic Memory Board provides 49,152 8-bit bytes of random access memory, using 24 16K dynamic memory chips. It can be used in ANY S-100 bus computer using a Z-80 CPU board. (Minor modifications may be necessary. See Section III, "User's Guide", below.). It occupies the lower 48K of memory, i.e. beginning at 0000H.

The Vector Graphic 48K Dynamic Memory Board is clearly a breakthrough in cost effectiveness. This is accomplished by combining compact inexpensive dynamic memory chips with the use of the refresh provisions of the Z-80 CPU. The most recent static RAM boards cost considerably more, take up more space, and require more power. Other dynamic memory boards use complex support logic on the board, rather than the built in features of the Z-80.

In addition to the above features, the Vector Graphic Dynamic Memory Board has proven to be remarkably reliable. Considerable attention was given during design of the board to the elimination of noise. It features a gridded ground plane designed to reduce noise. Accepted design practice was observed in structuring grounds, power supply, and bypass.

1.2 Description of the Manual

This manual provides a general description of the Vector Graphic Dynamic Memory Board, a more detailed discussion of the theory of operation, and information on how to use and test the board. Since the board is not sold as a kit, assembly information and parts list are not included.

II. THEORY OF OPERATION

2.1 The 48K Board Circuitry

The Vector Graphic 48K Dynamic Memory Board uses the refresh provisions of the Z-80 CPU board, which greatly simplifies the support logic circuitry on the board.

A memory reference cycle begins with the CPU board sending the address over A0-A15. Each of the memory chips contains 16K bits, so 14 address bits must be supplied to the chips. This is done by time multiplexing 7 address inputs. Initially the low order 7 addresses are applied to the chip address inputs by U39. After the address lines have had time to settle, the CPU board issues a memory read signal on SMEMR or a write signal in MWRITE. This signal propagates down a chain of inverters in U13 to initiate a timing sequence which does the following things in order:

1. Generates a low going $\overline{\text{RAS}}$ strobe for the appropriate memory block which is selected by U35 and gates U11 and U23. This strobe latches the address bits internally in the memory chips.

2. Disables U39, a tri state driver.

3. Enables U38, applying the high order 7 address lines to the chips.

4. Generates a low going $\overline{\text{CAS}}$ strobe which latches the addresses in the memory chips and initiates a timing sequence in the chip to enable the output drivers if it is a read cycle, or to write the data into the selected memory location if it is a write cycle.

If the cycle is a read cycle or M1 cycle, the logic associated with U43 enables the bus driver U41 to place the memory data on the DI bus to be read by the CPU. If the cycle is a write cycle, data on the DO bus is buffered by U40 and made available at the data input pin of the chips. The 74LS244 chips have Schmitt trigger inputs to discriminate against noise.

The principal difference between static and dynamic memory is the need to refresh the data stored in a dynamic memory. This is necessary because each bit of data is represented by an electric charge stored on a capacitor in each memory "cell". This charge will gradually leak away due to the finite insulation resistance of the dielectric. In order to restore the amplitude of the charge, it must be periodically read out, amplified, and written back in the same location. This is accomplished, thanks to the ingenuity of the chip designers, by executing a $\overline{\text{RAS}}$ only memory cycle for each of the 128 row addresses within a 2 millisecond interval. In other dynamic

memory boards, this is done using counters, multiplexers and complicated priority resolving circuitry to interleave refresh cycles with CPU access cycles.

2.2 The Refresh Feature of the Z-80 Board

Fortunately, the designers of the Z-80 provided for this requirement by including a 7 bit refresh counter in the chip which is incremented every M1 cycle and by designing the chip to output the count on the address bus during every M1 cycle at a point in time when the address bus is idle. The Vector Graphic Z-80 board outputs a RFSH signal on pin 66 of the bus to indicate that a refresh cycle must be initiated. The timing of this signal is modified by U24 and reclocked by the system clock to satisfy the timing requirements of the memory chips. The output of U23 pin 8 is combined in U11 to generate a RAS strobe which is applied to all chips during the refresh cycle.

There are several conditions under which the normal refresh sequence will be interrupted which could result in loss of memory data.

1. if the CPU is held in a wait state for 2 mS or longer.

2. if a DMA device takes control of the bus for more than 2 mS without generating the necessary refresh signals.

3. if the CPU is held in a reset condition for more than 2 mS.

Condition 1 would not occur in a Vector Graphic computer, but can occur in a system with a front panel where the READY line is used to HALT program execution. There are other ways in which this can occur since the READY line is a popular way of suspending CPU operation. Caution should be exercised with PROM programming boards, which typically hold the CPU in a wait state of 600 micro-seconds during a programming pulse. See Section III, "User's Guide," for information on handling this problem with PROM programmers. Other common uses of the READY line are to synchronize disk transfers and to prevent video display glitching. These last two are usually not a problem due to the relatively short wait period, on the order of a few microseconds.

Condition 2 does not occur in a Vector Graphic computer since none of the boards use DMA. If you are using DMA along with the 48K board, the DMA must not interrupt the CPU for longer than 2 ms without generating the necessary refresh signals.

Condition 3 has been eliminated by providing additional circuitry on the memory board to generate a short reset pulse synchronized with a M1 cycle. This consists of an RC network C8, R12, R11, to condition the PRESET signal connected to the front panel reset switch. Grounding PRESET allows U1 to be set by the next M1 cycle (memory

cycles can not be interrupted without loss of data). U1 triggers U2 generating a 200 micro-second pulse to pull unused bus line 55 low through the open collector inverters in U14. If you purchased the 48K board as a separate board, not as part of a complete Vector Graphic computer, you will have to make certain minor modifications to your Z-80 CPU board so that the Reset circuit responds to the signal coming from the 48K board on pin 55. These modifications are described in the User's Guide (Section III of this manual.) Vector Graphic computers shipped with the 48K board will have the modifications made at the factory.

III. USER'S GUIDE

3.1 Modifications to the 48K Board and Z-80 Board

This section is concerned with making modifications to the jumpers on the 48K board and CPU board in order to use the 48K board in existing systems. If you are going to use the board in an existing system THIS SECTION IS VERY IMPORTANT. However, if the 48K board was shipped as a part of a complete Vector Graphic computer such as the MZ, MEMORITE, or Vector 3, skip this section.

1. REFRESH SIGNAL - This modification is only needed for certain non-Vector Graphic Z-80 CPU boards. It involves installing jumpers to provide the proper RFSH signal, making use of the signal presently coming from your CPU. This is to ensure that a refresh cycle is generated during every instruction fetch cycle. Without this signal, operation is not possible.

2. RESET CIRCUITRY - This modification is essential in almost all existing computer systems, INCLUDING EXISTING VECTOR GRAPHIC COMPUTERS. The only exceptions are those computers that do not use the PRESET line (pin 75 of the bus) to reset the CPU. The modification involves changing the reset circuit of the CPU board so that it responds to the short reset pulse generated by the 48K RAM board on pin 55 rather than responding directly to the PRESET signal on pin 75. This modification prevents the CPU from being held in a reset state for longer than 2 milliseconds which would result in loss of memory data, since the memory is not being refreshed while the CPU is in a reset state.

The above two modifications are discussed in detail below:

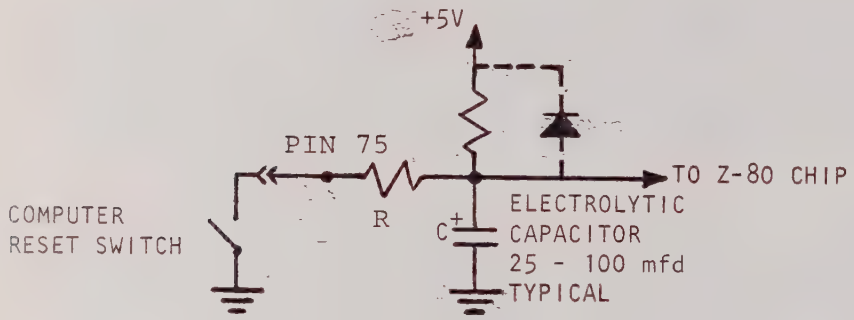
3.1.1 Refresh Signal

If you are using the 48K board with a non-Vector Graphic CPU board, refer to the manual for your CPU board to determine if the polarity and pin connection of the RFSH signal are correct. This signal is generated by pin 28 of the Z-80 CPU chip and should be buffered onto pin 66 of the bus using a non-inverting buffer of the 8097 or 74367 type. If this is not the case, it is necessary to modify the jumper arrangement on the 48K board. Area C on the board allows for inverting the polarity of the RFSH signal if it is inverted on the CPU board. Cut the trace between pad 2 and pad 1 and connect a jumper between pad 3 and pad 1. If the RFSH signal is brought to pin 98 on the bus instead of pin 66, a pad is provided on the 48K board. Cut the trace in area A to pin 66 and install a jumper to pin 98.

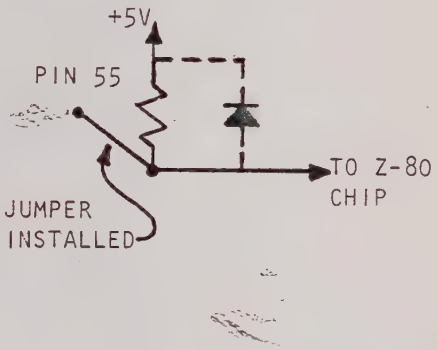
3.1.2 Reset Circuitry

The purpose of this modification is to allow the short (about 200 microseconds) pulse generated by the 48K board on pin 55 (previously unused) to reset the CPU, rather than using the PRESET signal generated on pin 75 by the Reset switch. The short signal on pin 55 is generated by the 48K board in response to the PRESET signal.

The modification is done to the Z-80 CPU board, NOT the 48K board. A typical Reset circuit on a Z-80 CPU board and the necessary modifications are shown below:



ORIGINAL CIRCUIT



MODIFIED CIRCUIT

Remove the Resistor R and Capacitor C from the board. Then connect a jumper from pin 55 to the pad previously connected to the + end of the capacitor.

The above gives the general idea. The specific modifications for each of four particular Z-80 boards are given below. If your Z-80 board is not one of these, then you will have to use the general description and drawing given above to determine how to modify it.

3.1.2.1 Vector Graphic Revision 1 Z-80 Boards

Install a jumper between pin 55 and the junction of the 220 ohm and 180 ohm resistor connected to the emitter of the 2N3643 transistor. This junction will be found near the upper right-hand corner of the schematic. Then, remove the 100 ohm resistor and the capacitor which are connected to pin 75.

3.1.2.2 Vector Graphic Revision 2 and later Z-80 Boards

The relevent circuitry is the same as Revision 1, but jumper pads are provided for this modification. Simply cut the trace in Area D and tie pad E to pad F. These pads are all found at the bottom edge

of the board.

3.1.2.3 North Star Z-80 Board

1) Remove capacitor C21
2) Remove resistor R17
3) Connect a jumper from IC "7G" pin 5 to pin 55 of the S-100 edge connector.

No modifications for the $\overline{\text{RFSH}}$ signal are needed for the Northstar Z-80 board.

3.1.2.4 Cromemco "ZPU" Board

1) Bend pin 11 of U11 out of the way, and then return U11 to its socket.
2) Cut the foil going to pin 75 on the S-100 edge connector.
3) Install a jumper from U11 pin 10 to pin 55 of the S-100 edge connector.

For Cromemco "ZPU" boards used in non-Cromemco systems that use Power-on-clear ($\overline{\text{POC}}$) for RESET (e.g. Vector Graphic systems, North Star systems, etc.) perform the following steps in addition to the above.

4) Bend pin 9 of U23 out of the way, and then return U23 to its socket.
5) Install a jumper from U11 pin 9 to pin 99 of the S-100 edge connector.

No modifications for the $\overline{\text{RFSH}}$ signal are needed for the Cromemco ZPU.

4.3.2 The READY Line

If the CPU is held in a wait state for 2 mS or longer, the normal refresh sequence will be interrupted which could result in the loss of memory data. Since computers with a front panel use the READY line to HALT program operation, if your computer has a front panel, you may NOT use the front panel when the 48K Board is being used. If you do, the system will bomb.

Caution should also be exercised with PROM programming boards, which typically hold the CPU in a wait state of 600 micro-seconds during a programming pulse. After every byte programmed, a software loop should execute 128 instructions to ensure refresh. A typical loop is as follows:

```
        PUSH B
        MVI  B,80H
LOOP    DJNZ LOOP
```

```
POP  B
```

Other common uses of the READY line are to synchronize disk transfers and to prevent video display glitching. These last two are usually not a problem due to the relatively short wait period, on the order of a few microseconds.

3.3 Using DMA

If you are using DMA along with the 48K board, the DMA must not interrupt the CPU for longer than 2 mS without generating the necessary refresh signals.

3.4 Address of the Board

The 48K RAM board is designed to occupy the lower 48K of address space. No provision has been made to change this. For this reason, you will not find any references in this manual to the method of specifying the address of the board.

3.5 Memory test

This section is relevent if you are using a Vector Graphic computer, or at least a Vector Graphic 12K PROM/RAM board in addition to the 48K board.

The 48K board can be tested using the Vector Graphic Extended Monitor T command. Install the board in your system and turn the system on. Type T 0000 C000. After a few seconds C000 XX C3 should appear on the terminal indicating that location C000 (the first PROM location) could not be written to. The test will automatically repeat, and no addresses other than C000 should be printed out. Depress the RESET key on the front panel to terminate the test.

A more thorough test can be made using the MDIAG program supplied on the Vector MZ system diskette. Refer to Appendix J of the MZ User's Manual.

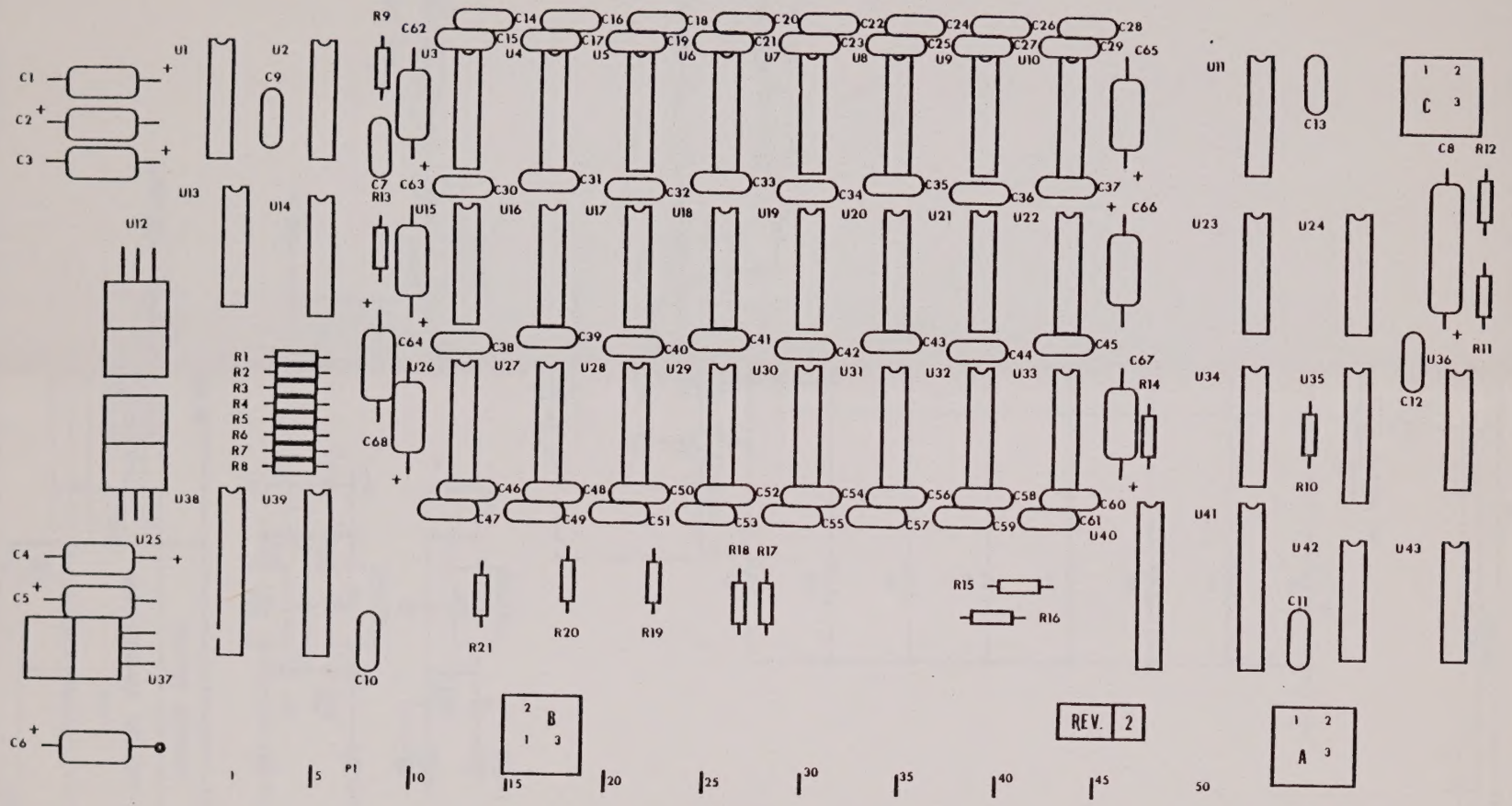
SPECIFICATIONS

Capacity:	49,152 8-bit bytes
Buffering:	Buffered data input and output
Access Time:	Compatible with Z-80 at 4MHZ without wait states
Power Consumption:	0.25A of +8V and 0.20A of +16V, typical
$\overline{\text{Phantom}}$:	Ouput buffer disable compatible with Vector Graphic PROM/RAM (Reset and Go) Board
DMA:	Must not interrupt the CPU for longer than $2X10^{-3}$ seconds without generating the necessary refresh signals
Availability:	Shipped assembled, tested, burned in, and guaranteed 1 year; no kits

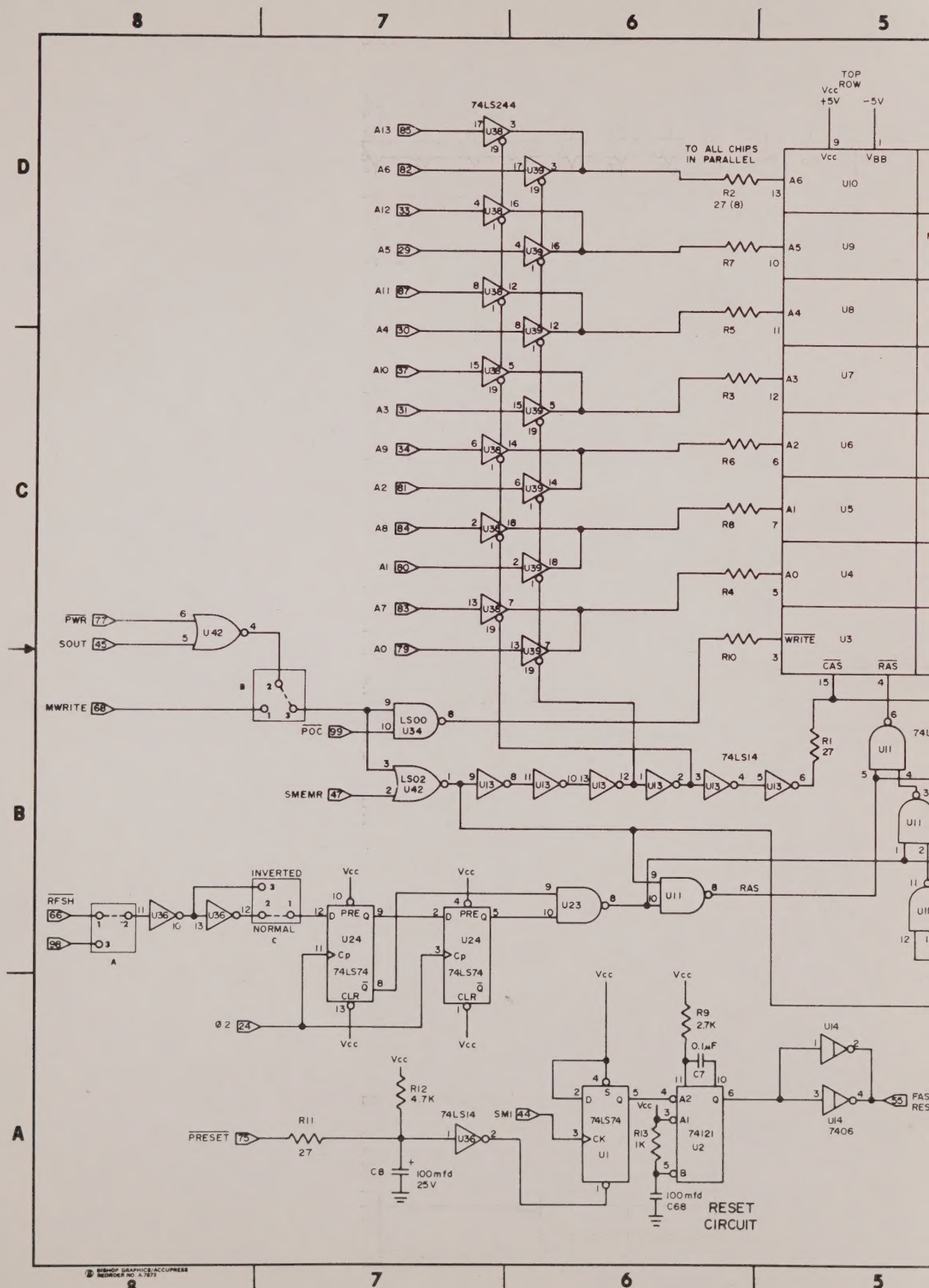
D7	D6	D5	D4	D3	D2	D1	D0	
U3							U10	0-3FFF
U15							U22	4000-7FFF
U26							U33	8000-BFFF

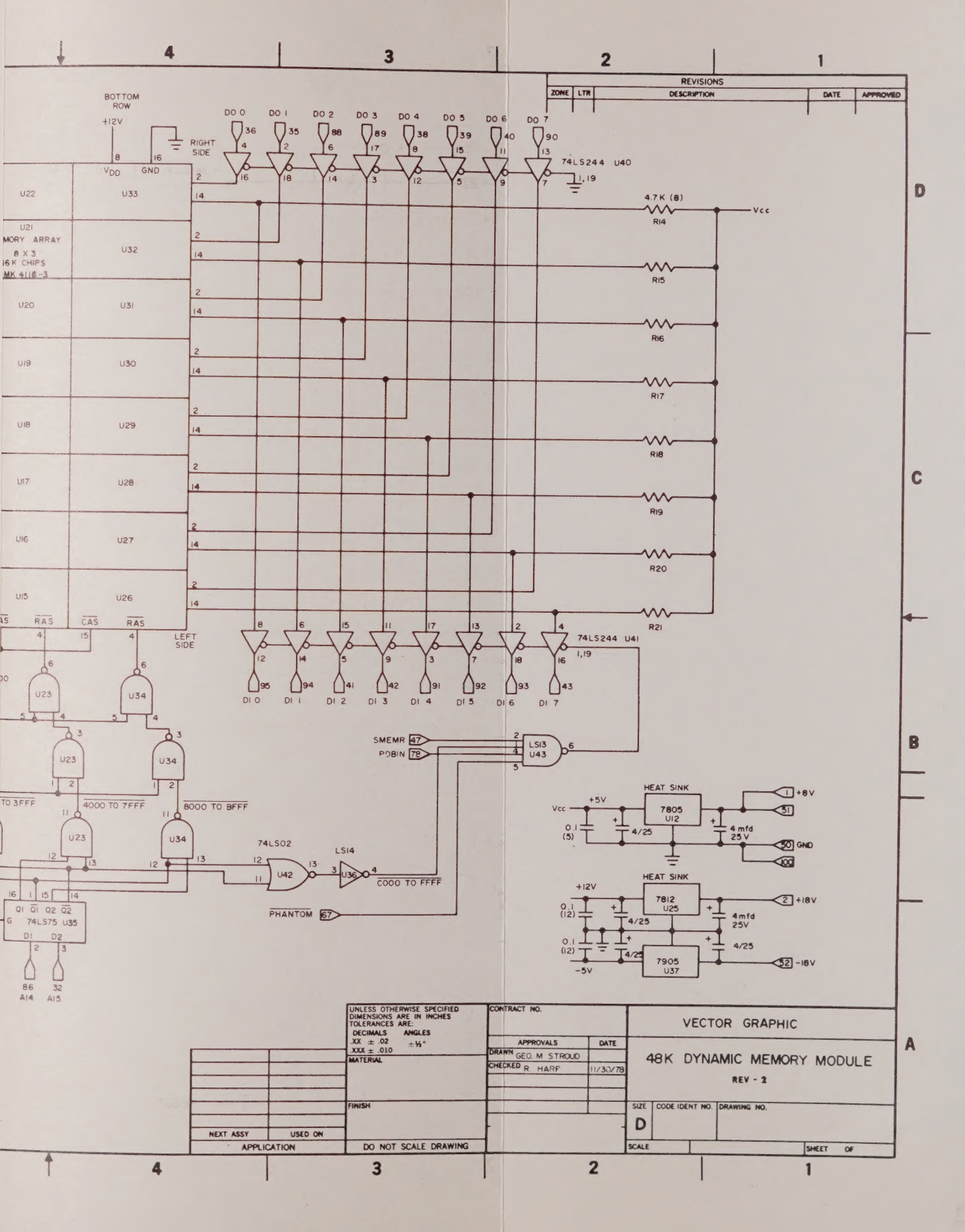
48K DYNAMIC MEMORY CHIP LOCATIONS

REDUCE TO 6.000 INCHES



SILKSCREEN
48K RAM


8
7
6
5
D
C
B
A
74LS244
TOP ROW
Vcc +5V
-5V
TO ALL CHIPS IN PARALLEL
A13 85
A6 82
A12 33
A5 29
A11 87
A4 30
A10 37
A3 31
A9 34
A2 81
A8 84
A1 80
A7 83
A0 79
R2 27 (8)
R7
R5
R3
R6
R8
R4
R10
A6 U10
A5 U9
A4 U8
A3 U7
A2 U6
A1 U5
A0 U4
WRITE U3
CAS
RAS
PWR 77
SOUT 45
U42
MWRITE 68
POC 99
LS00 U34
LS02 U42
SMEMR 47
74LS14
R1 27
U11
RAS
INVERTED
NORMAL
RFSH 66
98
U36
U23
PRE Q
U24
74LS74
CLR
Ø2 24
R9 2.7K
0.1mF
C7
U14
7406
R12 4.7K
R11
27
PRESET 75
SMI 44
U1
R13 1K
74121
U2
C8 100mfd 25V
100mfd C68
RESET CIRCUIT


REVISIONS
ZONE | LTR | DESCRIPTION | DATE | APPROVED
BOTTOM ROW
+12V
RIGHT SIDE
DO 0 DO 1 DO 2 DO 3 DO 4 DO 5 DO 6 DO 7
74LS244 U40
4.7K (8)
R14 R15 R16 R17 R18 R19 R20 R21
Vcc
U22 U33
U21 MORY ARRAY 8 X 3 16K CHIPS MK 4116-3
U32 U20 U31 U19 U30 U18 U29 U17 U28 U16 U27 U15 U26
RAS CAS RAS
LEFT SIDE
74LS244 U41
DI 0 DI 1 DI 2 DI 3 DI 4 DI 5 DI 6 DI 7
U23 U34 U23 U34 U23 U34
TO 3FFF
4000 TO 7FFF
8000 TO BFFF
SMEMR 47
PDBIN 78
LS13 U43
74LS02
LS14
U42 U36
COOO TO FFFF
PHANTOM 67
74LS75 U35
86 A14
32 A15
HEAT SINK
+5V
7805 U12
0.1 (5)
4/25
4 mfd 25V
+8V
GND
HEAT SINK
+12V
7812 U25
+18V
7905 U37
-5V
-18V
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE:
DECIMALS ANGLES
.XX ± .02 ±½°
.XXX ± .010
MATERIAL
FINISH
NEXT ASSY USED ON
APPLICATION
DO NOT SCALE DRAWING
CONTRACT NO.
APPROVALS DATE
DRAWN GEO. M STROUD
CHECKED R. HARP 11/30/78
VECTOR GRAPHIC
48K DYNAMIC MEMORY MODULE
REV - 2
SIZE D
CODE IDENT NO.
DRAWING NO.
SCALE
SHEET OF